НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК _____

«До захисту допущено»
В.о.завідувача кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)

“____” _____ 20 *19* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *126 Інформаційні системи та технології*
(код та назва спеціальності)

ОПП *Інформаційні управляючі системи та технології*
(код та назва спеціалізації)

на тему: *Інтелектуалізація обчислень для задач розрахунку стійкості конструкцій*

Виконав: студент *VI* курсу групи ІС-82мп
(шифр групи)

Богурський Денис Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *член кор. НАНУ, д.ф-м.н.,проф. Хіміч О.М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *доцент, к.т.н., доц. Жданова О.Г.* _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент *доцент каф. ТК, к.т.н., доцент Лісовиченко О.І.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

РЕФЕРАТ

Магістерська дисертація: 96 с., 16 рисунок, 11 табл., 1 додаток, 31 джерел.

Цілком очевидно, що адекватний опис та дослідження такого складного явища як втрата стійкості в конструкціях не можуть бути достовірно реалізовані в рамках двомірних прикладних теорій стійкості тонкостінних елементів (пластин, оболонок і т. д.).

Наявність високопродуктивної сучасної обчислювальної техніки та новітніх комп'ютерних технологій дають можливість перейти до розгляду тривимірних математичних моделей процесів розрахунку стійкості конструкцій та споруд, деформування та руйнування твердого тіла, хімічних процесів тощо. Очевидно, що розгляд проблем в такій постановці приводить до моделей великих розмірів, які потребують значних комп'ютерних ресурсів та нових ефективних алгоритмів розв'язування задач. Математичні моделі багатьох інженерних задач описуються системами диференціальних рівнянь або різницевими рівняннями, розв'язання яких полягає у визначенні власних значень і власних векторів матриць, що, як правило, мають розріджену структуру. І дуже часто це є однією з фундаментальних і ресурсномістких задач. Від ефективності розв'язування саме АПВЗ в значній мірі залежить ефективність розв'язування всієї проблеми.

Характерною їх особливістю матриць в цих задачах є надвеликі порядки (до десятків мільйонів), а кількість ненульових елементів складає kn , $де\ k << n$, n – порядок матриці. Отже, проблема створення ефективних алгоритмів розв'язання АПВЗ розріджених матриць на комп'ютерах гібридної архітектури є досить **актуальною**.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на філії кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Інтелектуалізація обчислень для задач розрахунку стійкості конструкцій».

Метою даної роботи є прискорення проектування конструкцій за допомогою автоматичного вибору алгоритму розв'язання задачі та застосування графічних

процесорів для прискорення обчислень.

Досягнення мети базується на розробці нейронної мережі для визначення типу вхідної матриці, розробці гібридних алгоритмів для розв’язування задачі алгебраїчної проблеми власних значень (АПВЗ), до якої зводиться задача моделювання стійкості конструкцій, а також на аналізі оцінок ефективності та прискорення розроблених гібридних алгоритмів.

Об’єктом даного дослідження є процес класифікації вхідних матриць розрідженої структури та розв’язання часткової узагальненої АПВЗ для стрічкових симетричних матриць великого розміру за допомогою гібридного алгоритму методу ітерацій на підпросторі.

Предмет дослідження – паралельні методи та комп’ютерні алгоритми знаходження розв’язку АПВЗ з розрідженими матрицями нерегулярної структури.

Наукова новизна: наукова новизна полягає у використанні нейронної мережі для класифікації типу вхідної матриці, що дозволяє використати оптимальний алгоритм розв’язання задачі, що сприяє раціональному використанню комп’ютерних ресурсів та змешенню загального часу знаходження розв'язку. Також новизна даної роботи полягає у розробленні гібридного алгоритму, який передбачає використання графічних процесорів для ресурсозатратних обчислень при роз'язанні, що прискорює вирішення задачі визначення стійкості конструкцій.

Публікації: За матеріалами дисертації було опубліковано 3 наукові роботи: 1 стаття та 2 тез доповідей на конференціях.

НЕЙРОННА МЕРЕЖА, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, ГІБРИДНІ АЛГОРИТМИ, АЛГЕБРАЇЧНА ПРОБЛЕМА ВЛАСНИХ ЗНАЧЕНЬ (АПВЗ), КЛАСИФІКАЦІЯ ДАНИХ, РОЗРІДЖЕНІ МАТРИЦІ

ABSTRACT

Master's Thesis: 96 pp., 16 figs., 11 tables, 1 appendix, 31 sources.

It is quite obvious that adequate description and investigation of such a complex phenomenon as loss of stability in structures cannot be reliably realized within the framework of two-dimensional applied theories of the stability of thin-walled elements (plates, shells, etc.).

The presence of high-performance modern computer technology and the latest computer technologies make it possible to move to the consideration of three-dimensional mathematical models of the processes of calculating the stability of structures and structures, deformation and destruction of solid bodies, chemical processes, etc. Obviously, addressing problems in this formulation leads to large-scale models that require significant computer resources and new efficient problem solving algorithms. The mathematical models of many engineering problems are described by systems of differential equations or difference equations whose solution is to determine the eigen values and eigen vectors of the matrices, which usually have a sparse structure. And very often this is one of the fundamental and resource-intensive tasks. The efficiency of solving the whole problem depends largely on the resolution of the APEV.

Their characteristic feature of matrices in these problems is very large orders of magnitude (up to tens of millions), and the number of non-zero elements is kn, where $k << n$, n - is the order of the matrix. Therefore, the problem of creating efficient algorithms for decomposing the AHP of sparse matrices on hybrid architecture computers is quite urgent.

The purpose of this work is to accelerate the design of structures by automatically selecting the algorithm for solving the problem and using GPUs to accelerate the calculations.

Relationship with working with scientific programs, plans, topics. Work performed at the branch of the Department of Automated Information Processing and Management Systems of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky» within the topic «Intellectualization of calculations for structural stability calculation problems».

The goal is based on the development of a neural network to determine the type of

input matrix, the development of hybrid algorithms for solving the problem of algebraic eigenvalue problem (APEV), which reduces the problem of modeling the stability of structures, as well as the analysis of the estimates of the efficiency and acceleration of developed hybrids.

The object of this study is mathematical models that describe SLAE with sparse matrices of irregular structure.

The subject of the study are parallel methods and computer algorithms for finding the SLAE solution with sparse matrices of irregular structure.

Scientific Novelty: The scientific novelty is to use a neural network to classify the type of input matrix, which allows us to use the optimal algorithm for solving the problem, which contributes to the rational use of computer resources and to reduce the total time of finding the solution. Also, the novelty of this work is the development of a hybrid algorithm that involves the use of graphics processors for resource-intensive computations when cutting, which speeds up the solution of the problem of determining the stability of structures.

Publications: Based on the dissertation materials, 3 scientific papers were published: 1 article and 2 abstracts at conferences.

NEURAL NETWORK, IMAGE RECOGNITION, HYBRID ALGORITHMS, ALGEBRAIC PROBLEM OF EIGENVALUES (APEV), CLASSIFICATION OF DATA, SPARSE MATRICES

ЗМІСТ

ВСТУП

Стійкість – це здатність тіл протидіяти зовнішнім силам, зберігаючи первісну форму пружної рівноваги.

Якщо при дії малих збурень тіло відхиляється від свого незбуреного стану рівноваги незначно, а після припинення дії малих збурень повертається у вихідний стан, то такий стан називається стійким. Якщо ж стан рівноваги тіла не володіє цією властивістю, тобто після припинення дії малих збурень тіло не повертається в початковий стан, то такий стан називається нестійким.

Наслідком втрати стійкості зазвичай є обвалення конструкції, частини або всієї споруди. Необхідність забезпечення стійкості конструкцій - одна з основних вимог норм проектування.

Безліч практичних та наукових задач, зокрема, при дослідженні стійкості конструкцій, розрахунку динаміки напружено-деформованого стану об’єктів різної природи та ін., зводяться до розв’язання часткової алгебраїчної проблеми власних значень стрічкових симетричних додатно-означених матриць великої розмірності. Застосування гібридних комп’ютерів для розв’язування таких задач потребує створення алгоритмів, які враховують унікальні архітектурні та обчислювальні особливості цих комп’ютерів.

Метою даної роботи є прискорення проектування конструкцій за допомогою автоматичного вибору алгоритму розв’язання задачі та застосування графічних процесорів для прискорення обчислень.

Досягнення мети базується на розробці нейронної мережі для визначення типу вхідної матриці, розробці гібридних алгоритмів для розв’язування задачі алгебраїчної проблеми власних значень (АПВЗ), до якої зводиться задача моделювання стійкості конструкцій, а також на аналізі оцінок ефективності та прискорення розроблених гібридних алгоритмів.

Об’єктом даного дослідження є процес класифікації вхідних матриць розрідженої структури та розв’язання часткової узагальненої АПВЗ для стрічкових симетричних матриць великого розміру за допомогою гібридного алгоритму методу

ітерацій на підпросторі.

Предмет дослідження – паралельні методи та комп’ютерні алгоритми знаходження розв’язку АПВЗ з розрідженими матрицями нерегулярної структури.

Наукова новизна: наукова новизна полягає у використанні нейронної мережі для класифікації типу вхідної матриці, що дозволяє використати оптимальний алгоритм розв’язання задачі, що сприяє раціональному використанню комп’ютерних ресурсів та зменшенню загального часу знаходження розв'язку. Також новизною даної роботи полягає у розробленні гібридного алгоритму, який передбачає використання графічних процесорів для ресурсозатратних обчислень при розв'язанні, що прискорює вирішення задачі визначення стійкості конструкцій.

Математичне моделювання процесів, пов’язаних з дослідженнями стійкості конструкцій, виникає в різних предметних областях. На сьогодні розробка моделей і методів дослідження властивостей композиту та його компонентів відіграють велику роль у проектуванні та дослідженні процесів деформування і руйнування конструкцій, що складаються з композиційних матеріалів.

Математичні моделі багатьох інженерних задач описуються системами диференціальних рівнянь або різницевими рівняннями, розв’язання яких полягає у визначенні власних значень і власних векторів матриць, що, як правило, мають розріджену структуру. Характерною їх особливістю є надвеликі порядки (до десятків мільйонів), а кількість ненульових елементів складає kn , де $k << n$, n – порядок матриці. Отже, проблема створення ефективних алгоритмів розв’язання АПВЗ розріджених матриць на комп’ютерах гібридної архітектури є досить актуальною.

Композитом називається неоднорідний матеріал, що складається з двох або кількох компонент (фаз). Наприклад, сюди можуть належати різні сплави, бетони, метали, полікристалічні тіла тощо. В механіці, заради зручності, одну з фаз називають матрицею (або елементом), а інші – включеннями (або армуючими елементами).

Композиційні матеріали, зазвичай, класифікуються за формою включень. Найпоширенішими серед них є макрочастини (гранули), короткі (або розірвані) волокна, безперервні довгі волокна (нитки), а також шари. Матриця забезпечує монолітність композиту; фіксує форму виробу і взаємне розташування армуючих

волокон; розподіляє діючі напруження за обсягом матеріалу, забезпечуючи рівномірне навантаження на волокно і її перерозподіл при руйнуванні частини волокон.

Розмір і геометрія області неоднорідності напруженого стану, характер розподілу напружень та деформацій залежать від співвідношення механічних і геометричних характеристик компонент матеріалу, ступеня їх анізотропії (відмінності властивостей середовища у різних напрямках.) а також від співвідношення геометричних параметрів, які визначають мікроструктуру композиту, розмір розрахункової області. Наявність в області поверхневого навантаження відмінних від нуля зсувних або дотичних напружень може істотно вплинути на критичні параметри стійкості композитного матеріалу [1 – 3]. При експлуатаційних навантаженнях і технологічних процесах виготовлення волокнистих і шаруватих композитних матеріалів виникають стискаючі напруження, які можуть призвести до втрати стійкості в структурі композитного матеріалу.

1 ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ

Цілком очевидно, що адекватний опис та дослідження такого складного явища як втрата стійкості в структурі композитних матеріалів не можуть бути достовірно реалізовані в рамках двомірних прикладних теорій стійкості тонкостінних елементів (пластин, оболонок і т. д.).

Тут розглядається задача стійкості композитних матеріалів нескінченно довгих смуг (нескінченних в напрямку перпендикуляра до даної площини), які мають кінцеву ширину, тобто до шаруватих композитів з наповнювачем стрічкової структури.

Найбільш ефективним підходом розв'язування задач зазначеного класу є застосування моделі шматково-однорідного середовища та основних співвідношень тривимірної лінеаризованої теорії стійкості деформованих тіл (ТЛТСДТ), які враховують неоднорідність докритичного стану [2, 3].

Складність отримання аналітичних розв'язків таких задач передбачає дослідження тривимірних моделей, використання сучасних чисельних методів [3 – 5] та потужних комп'ютерів для їх реалізації. При цьому найбільші комп'ютерні ресурси витрачаються на розв'язання задачі алгебраїчної проблеми власних значень (АПВЗ), до якої зводиться задача моделювання стійкості композитних матеріалів.

В наш час існує низка бібліотек програм, до яких увійшли програми для розв'язування АПВЗ розріджених матриць на різних комп'ютерних архітектурах: PETSc, SLEPc, Lis, NAG тощо. Серед програмних засобів, призначених для гібридних комп'ютерів, ефективні реалізації алгоритмів пропонують бібліотеки cuSOLVER, Magma.

Як показує огляд літератури та досвід використання існуючого алгоритмічного та програмного забезпечення з лінійної алгебри, тільки за умови правильного використання особливостей структури матриці та архітектурних особливостей комп'ютера можна створити ефективні гібридні алгоритми розв'язування АПВЗ для розріджених матриць.

1.1 Висновки до розділу

У даному розділі було подано огляд проблеи стійкості конструкцій, виділено найбільш ефективний підхід розв’язування задач зазначеного класу, наведено кілька бібліотек для розв’язання такого типу задач.

2 ПОСТАНОВКА ТА ФОРМАЛІЗАЦІЯ ЗАДАЧІ

Розв’язування багатьох практичних задач методом скінченних елементів або скінченних різниць для диференціальних рівнянь (наприклад, дослідження стійкості конструкцій) зводиться до часткової узагальненої АПВЗ для стрічкових симетричних матриць великого розміру. Для комп’ютерного розв’язування цих задач ефективним є метод ітерацій на підпросторі. У цьому розділі розглянуто математичне моделювання задачі стійкості композитних матеріалів, яка зведена до розв’язування АПВЗ методом ітерацій на підпросторі для розв’язування часткової узагальненої АПВЗ (кілька власних значень) стрічкових симетричних матриць, одна з яких додатно визначена. Подано результати розв’язування задачі стійкості композитного матеріалу при використанні алгоритмів методу ітерацій на підпросторі для комп’ютерів різної архітектури.

Тут розглядається задача стійкості композитних матеріалів нескінченно довгих смуг (які є нескінченними в напрямку перпендикуляра до даної площини) кінцевої ширини, тобто до шаруватих композитів з наповнювачем стрічкової структури.

Найбільш ефективним підходом розв’язування задач зазначеного класу є застосування моделі шматково-однорідного середовища та основних співвідношень тривимірної лінеаризованої теорії стійкості деформованих тіл (ТЛТСДТ), які враховують неоднорідність докритичного стану [1 – 3].

Складність отримання аналітичних розв’язків таких задач передбачає дослідження тривимірних моделей, використання сучасних чисельних методів [3 – 5] та потужних комп’ютерів для їх реалізації. При цьому найбільші комп’ютерні ресурси витрачаються на розв’язування задачі АПВЗ, до якої зводиться задача моделювання стійкості композитних матеріалів.

2.1 Змістовна постановка задачі

Розглядається загальна постановка задачі визначення докритичного стану та задачі стійкості композитного матеріалу в рамках моделі «волокон кінцевих розмірів». Дослідження проводяться в рамках двомірної моделі (плоска деформація) в площині $x_1 0 x_2$ із застосуванням основних співвідношень ТЛТСДТ для малих

докритичних деформацій, а також моделі пружного ізотропного тіла для армуючих елементів і матриці. Застосовуються лагранжеві координати, які в недеформованому стані співпадають з декартовими координатами.

На рисунку 2.1 приведена розрахункова схема для найпростішого випадку, коли волокна не взаємодіють між собою через матрицю як в докритичному стані, так і при втраті стійкості. Зазначена ситуація виникає для композитних матеріалів з досить малою концентрацією наповнювача.

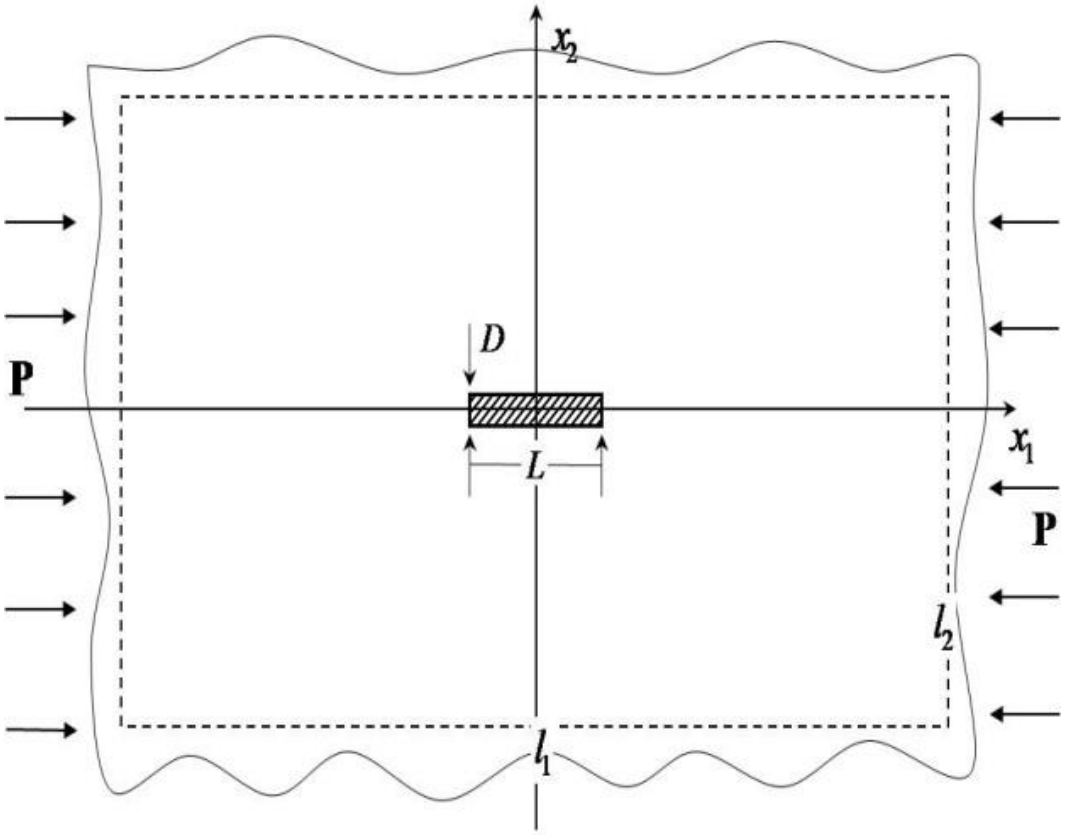

Рисунок 2.1 - Розрахункова схема задачі стійкості композитного матеріалу

Дослідження докритичного стану виконується в рамках класичної лінійної теорії пружності ізотропного тіла, рівняння рівноваги і співвідношення пружності якої можна представити в наступному вигляді:

$$\frac{\partial}{\partial x_i}\sigma_{ij}^0 = 0; \quad \sigma_{ij}^0 = \delta_{ij}\lambda\varepsilon_{nn}^0 + 2\mu\varepsilon_{ij}^0; \quad 2\varepsilon_{ij}^0 = \frac{\partial u_i^0}{\partial x_j} + \frac{\partial u_j^0}{\partial x_i} \qquad (2.1)$$

Тут і далі індексом “0” позначаються величини, що відносяться до початкового (докритичного) стану, індексами a та m – величини, що відносяться до матеріалу волокна і матриці відповідно.

Оскільки для матриці дослідження (2.1) проводиться для нескінченної області, то напруження і переміщення в матриці зручно представити у вигляді суми компонентів напружень і переміщень (2.2), що відповідають зовнішньому навантаженню, заданого для матриці «на нескінченності», та компонентів відповідних збурень напружено-деформованого стану, зумовленими наявністю волокна кінцевих розмірів:

$$\sigma_{ij}^{0\,m} = \sigma_{ij}^{\infty} + \sigma_{ij}^{10\,m}, \quad u_j^{0\,m} = u_j^{\infty} + u_j^{10\,m}. \tag{2.2}$$

Відповідні значення для σ та u визначаються так:

$$\sigma_{11}^{\infty} = -P,\ \sigma_{22}^{\infty} = 0,\ \ \sigma_{12}^{\infty} = 0,\ \ u_1^{\infty} = A_1 x_1,\ u_2^{\infty} = A_2 x_2. \tag{2.3}$$

Визначення докритичного стану проводиться з використанням зазначених величин та основних співвідношень, які включають такі умови безперервності векторів напружень і переміщень на межах розділення компонентів композиту:

$$\sigma_{11}^{\infty} + \sigma_{11}^{10m} = \sigma_{11}^{0\,a},\ \sigma_{12}^{10m} = \sigma_{12}^{0\,a},\ u_j^{\infty} + u_j^{10m} = u_j^{0a},\ x_1 = \pm {}^{L}/_{2},$$
$$|x_2| \le \pm {}^{D}/_{2}, \tag{2.4}$$

$$\sigma_{22}^{10m} = \sigma_{22}^{0\,a},\ \sigma_{12}^{10m} = \sigma_{12}^{0\,a},\ \ u_j^{\infty} + u_j^{10m} = u_j^{0a},\quad |x_1| \le \pm {}^{L}/_{2},$$
$$x_2 = \pm {}^{D}/_{2}, \tag{2.5}$$

а також умов загасання «на нескінченності»:

$$\sigma_{ij}^{10\,m} \to 0,\ u_j^{10\,m} \to 0 \qquad \text{при} \quad \sqrt{x_1^2 + x_2^2} \to \infty. \tag{2.6}$$

Дослідження задачі стійкості виконується із застосуванням статичного методу ТЛТСДТ в рамках другого варіанту теорії малих докритичних деформацій при моделюванні наповнювача і матриці лінійно-пружним ізотропним тілом, що узгоджується з постановкою задачі визначення докритичного напружено-деформованого стану.

Таким чином, рівняння стійкості і складові несиметричного тензора напружень можна представити в наступному вигляді:

$$\frac{\partial}{\partial x_i}\left(\omega_{ij\alpha\beta}\frac{\partial}{\partial x_\beta}u_\alpha\right) = 0, t_{ij} = \omega_{ij\alpha\beta}\frac{\partial}{\partial x_\beta}u_\alpha, i,j,\alpha,\beta = 1{,}2. \tag{2.7}$$

Для компонентів тензора ω використовуються такі формули:

$$\omega_{ij\alpha\beta} = \delta_{ij}\delta_{\alpha\beta}\lambda + (\delta_{i\beta}\delta_{\alpha j} + \delta_{i\alpha}\delta_{\beta j})\mu + \delta_{\alpha j}\sigma_{i\beta}^{0}, \qquad \sigma_{i\beta}^{0} = \delta_{i\beta}\sigma_{\beta\beta}^{0}, \tag{2.8}$$

де λ, μ– постійні Ляме.

При дослідженні задачі стійкості співвідношення (2.7) застосовуються для матриці, записавши їх відносно величин ${\sigma_{ij}^1}^m$, ${\varepsilon_{ij}^1}^m$, ${u_j^1}^m$, ${\omega_{ij\alpha\beta}^1}^m$, λ_m, μ_m, а також – для волокна, записавши їх відносно величин ${\sigma_{ij}}^a$, ${\varepsilon_{ij}}^a$, ${u_j}^a$, ${\omega_{ij\alpha\beta}}^a$, λ_a, μ_a.

Таким чином, у відповідності до (2), (3) та (8) мають місце такі вирази для матриці і волокна відповідно:

$$ {\omega_{ij\alpha\beta}^1}^m = \delta_{ij}\delta_{\alpha\beta}\lambda_m + (\delta_{i\beta}\delta_{\alpha j} + \delta_{i\alpha}\delta_{\beta j})\mu_m + \delta_{\alpha j}{\sigma_{i\beta}^0}^m, $$

$$ {\sigma_{i\beta}^0}^m = -\delta_{i\beta}\delta_{\beta 1}P + \sigma_{i\beta}^{10m}; \quad (2.9) $$

$$ {\omega_{ij\alpha\beta}}^a = \delta_{ij}\delta_{\alpha\beta}\lambda_a + (\delta_{i\beta}\delta_{\alpha j} + \delta_{i\alpha}\delta_{\beta j})\mu_a + \delta_{\alpha j}{\sigma_{i\beta}^0}^a. \quad (2.10) $$

Повне формулювання задачі стійкості включає також умови безперервності векторів напружень та переміщень на межах розділення, які для описаної розрахункової схеми можна представити в наступному вигляді:

$$ {t_{11}^1}^m = {t_{11}}^a, \ {t_{12}^1}^m = {t_{12}}^a, \ {u_1^1}^m = {u_1}^a, \ {u_2^1}^m = {u_2}^a, \ x_1 = \pm {L}/{2}, $$

$$ |x_2| \le \pm {D}/{2}, \quad (12.11) $$

$$ {t_{22}^1}^m = {t_{22}}^a, \ {t_{21}^1}^m = {t_{21}}^a, \ {u_1^1}^m = {u_1}^a, \ {u_2^1}^m = {u_2}^a, \ |x_1| \le \pm {L}/{2}, \quad x_1 = \pm {D}/{2}, \quad (2.12) $$

а також умови загасання «на нескінченності», які для описаної розрахункової схеми мають такий вигляд:

$$ {u_j^1}^m \to 0, \quad \text{при} \quad \sqrt{x_1^2 + x_2^2} \to \infty. \quad (2.13) $$

Докритичний стан в ситуації, представленої на рисунку 1, відповідає задачі про концентрацію напружень (при осьовому навантаженні) біля прямокутного включення, яке заповнене матеріалом з різними характеристиками. Отже, докритичний стан буде істотно неоднорідним.

Таким чином, для плоскої деформації отримуємо задачу на власні значення для системи рівнянь з частинними похідними із змінними коефіцієнтами, які залежать від двох змінних – x_1, x_2. В такому випадку, отримати розв’язок відповідної задачі на власні значення з використанням аналітичних методів неможливо, що обумовлює необхідність застосування чисельних методів.

В процесі дослідження стійкості композитного матеріалу під дією стискаючого навантаження виконуються умови самоспряжених операторів задачі визначення початкового напружено-деформованого стану (2.1) – (2.6) і узагальненої задачі на власні значення, що відповідає задачі тривимірної теорії стійкості (2.7) – (2.13). Крім того, диференціальні оператори розглянутих задач є додатно визначеними за умови відсутності жорсткого зміщення.

Вибір методів розв'язування дискретних задач та збіжність обчислювальних процесів залежать від властивостей отриманих різницевих операторів. Тому різницеві задачі доцільно будувати так, щоб оператори зберігали ті ж властивості, що і самоспряжені оператори, а також – додатну визначеність диференціальних операторів.

2.2 Побудова дискретної моделі

Розглянемо методику чисельного розв'язування задачі тривимірної лінеаризованої теорії стійкості та задачі лінійної теорії пружності методом скінченних різниць із застосуванням концепції базових схем. Такий підхід детально описано для застосування до широкого класу задач механіки деформованого твердого тіла в роботі [4], а для задач тривимірної стійкості композитних матеріалів армованих волокнами кінчених розмірів – вперше запропончовано в роботі [5].

В якості основних математичних моделей приймаються рівняння лінійної теорії пружності шматково-однорідних середовищ для визначення докритичного напружено-деформованого стану (рівняння тривимірної лінеаризованої теорії стійкості при малих деформаціях для визначення критичних параметрів).

В якості механічої моделі розглядається модель шматково-однорідного середовища. Компоненти композиту моделюються лінійно-пружними ізотропними тілами.

Використання концепції базових схем лежить в основі запропонованого способу побудови **дискретних моделей**. Базова схема – це різницева схема, побудована на шаблоні комірки сітки, яка дозволяє виписати в явному вигляді систему сіткових рівнянь на шаблоні комірки сітки, що відповідає довільній задачі з розглянутого класу задач.

Завдяки підсумовуванню відповідних значень базових схем у кожному вузлі даної сіткової області ми отримуємо різницеву задачу, яка відповідає конкретній задачі з розглянутої множини.

Коротко розглянемо методику побудови дискретних моделей, при цьому обмежимося тільки випадком плоских задач. Для цього в розрахунковій області $\bar{\Omega}$ за допомогою прямих $x_i = const$ введемо нерівномірну по кожному напрямку x_i різницеву сітку так, щоб в межах комірки сітки матеріал був однорідним:

$$\bar{\omega} = \left\{x = (x_1^m, x_2^n),\ m = \overline{0, M},\ \ n = \overline{0, N}\right\}. \tag{2.14}$$

Таким чином, сіткова область $\bar{\omega} = \omega \cup \gamma$, де ω – множина внутрішніх, γ – це множина граничних вузлів, є сукупністю прямокутних комірок сітки, кожна з яких може бути наділена механічними і геометричними характеристиками компонента композиту, що міститься в комірці (рисунок 2.2-2.3)

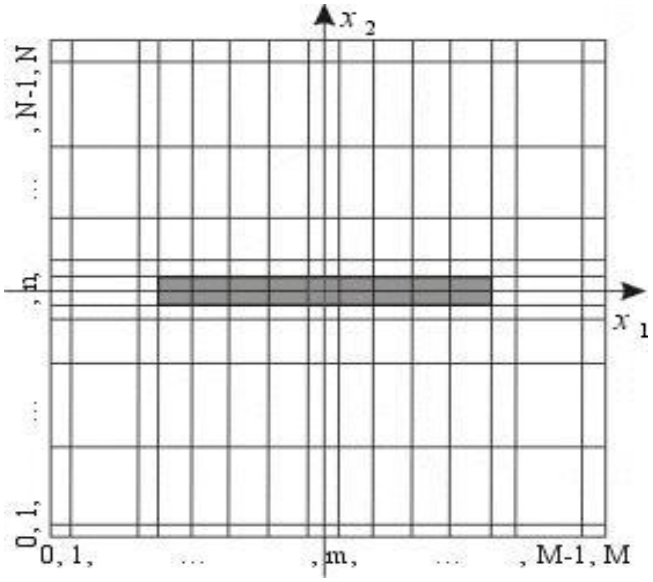

Рисунок 2.2 - Прямокутна комірка сітки

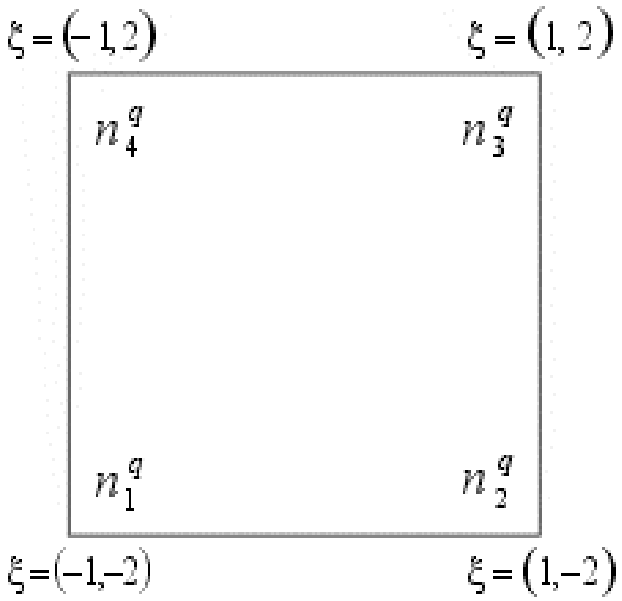

Рисунок 2.3 - Прямокутна комірка сітки

Виділяємо комірку сітки, пронумеруємо її вузли і у відповідність кожному вузлу поставимо векторний параметр $\boldsymbol{\xi} = (\xi_1, \xi_2) = (\pm 1, \pm 2)$. Очевидно, що до кожного вузла сітки $x \in \bar{\omega}$ примикає певна кількість комірок (від 1 до 4). До того ж комірки можуть відноситися до різних компонентів композитного матеріалу.

2.3 Побудова різницевої моделі

Побудова різницевої моделі дискретної задачі на сітці здійснюється варіаційно-різницевим способом, застосувавши концепції базових схем, компоненти яких можна визначити за допомогою апроксимації і мінімізації функціоналів, що відповідають задачам (2.1) – (2.6) та (2.7) – (2.13). При реалізації зазначеної процедури стосовно дослідження задачі стійкості використовуються варіаційні принципи ТЛТСДТ. Шляхом підсумовування значень базових схем у кожному вузлі сіткової області одержуються різницеві задачі, що є дискретними аналогами відповідних вихідних задач.

Операторну форму різницевої задачі визначення докритичного напружено-деформованого стану, яка відповідає задачі лінійної теорії пружності (1) – (6), можна записати в наступному вигляді:

$$A\,u = \Phi, x \in \overline{\omega} \quad \text{або} \quad A_i\,u = \Phi_i, x \in \overline{\omega}\ , \qquad (2.15)$$

де

$$A_i\, u = \sum_{\xi \in x} a_i(\xi) u$$

$$\Phi_i = \sum_{\xi \in x} \varphi_i\,(\xi) \quad x \in \overline{\omega} \qquad (2.16)$$

В (2.16) знак суми означає що у вузлі підсумовуються базові оператори для тих параметрів, які відповідають вузлу *x* у всіх суміжних до вибраного вузла комірках.

Вирази базових операторів, отримані на шаблоні комірки сітки, мають такий вигляд:

$$a_i(\xi)\, u = -\, H\, \frac{\sigma_{ji} + \sigma_{ji}^{\xi_j}}{\eta_{\xi_j}}$$

$$\phi\varphi_i(\xi) = -H\left(F_i^0 + \frac{2}{\eta_{\xi_j}} P_{ij}^0\right), x \in \gamma,$$

де $H = h_1 h_2$ – площа комірки сітки, $h_i = \eta_{\xi_i} sign\xi_i > 0$.

Різницевий оператор задачі (2.15) зберігає властивості самоспряженості та додатної визначеності відповідного диференціального оператора. Таким чином, задача визначення докритичного напружено-деформованого стану зводиться до розв'язування сіткових рівнянь (2.15), які можуть бути представлені у вигляді системи лінійних алгебраїчних рівнянь. Матриця системи, як правило, є симетричною та розрідженою, тобто складається в основному з нульових елементів, крім елементів, розміщених на декількох діагоналях.

Операторна форма різницевої задачі, відповідна диференціальній задачі стійкості (7) – (13), може бути представлена в наступному вигляді:

$$A\, u = p\, B\, u, \qquad x \in \overline{\omega} \;\text{ або }\; A_i\, u = p\, B_{i\,u} \quad x \in \overline{\omega}\,, \qquad (2.17)$$

де

$$A_i\, u = \sum_{\xi \in x} a_i(\xi) u\,, \qquad B_i\, u = \sum_{\xi \in x} b_i(\xi) u, \qquad x \in \overline{\omega}.$$

Вирази базових операторів, отримані на шаблоні комірок сітки, мають такий вигляд:

$$a_i(\xi)\,u = -\,H\,\frac{\sigma_{ji} + \sigma_{ji}^{\xi_j}}{\eta_{\xi_j}}, \qquad b_i(\xi)\,u = -\,H\,\frac{\sigma_{jk}^0 u_{i,\xi_k} + (\sigma_{jk}^0 u_{i,\xi_k})^{\xi_j}}{\eta_{\xi_j}},$$

де $H = h_1 h_2$ – площа комірки сітки, $h_i = \eta_{\xi_i} sign\xi_i > 0$.

Різницеві оператори задачі (2.17) зберігають властивості самоспряженості та додатної визначеності відповідних диференціальних операторів.

Таким чином, задача стійкості зводиться до розв'язування сіткових рівнянь (2.17), які можуть бути представлені у вигляді часткової узагальненої алгебраїчної задачі на власні значення.

При описаному способі побудови різницевих задач, завдяки збереженню властивостей диференціальних операторів, матриці відповідної часткової узагальненої задачі на власні значення A та B є симетричними, а також додатно визначеними при певних умовах, описаних вище стосовно класу задач, що розглядаються.

Крім того слід зазначити, що матриці A та B є розрідженими (стрічковими), тобто складаються в основному з нульових елементів, але без елементів, розміщених на декількох діагоналях.

2.4 Розв'язування часткової узагальненої АПВЗ задачі тривимірної теорії стійкості композитів

Аналіз задачі тривимірної теорії стійкості композитів, застосувавши до неї модель «волокон кінцевих розмірів» (рисунок 1) показав, що серед різницевих задач, до яких вона зводиться, розв'язування часткової узагальненої АПВЗ вимагає найбільших комп'ютерних ресурсів та часу розв'язування. Отже, ефективність розв'язування всієї задачі в основному залежить від ефективного розв'язування АПВЗ.

Тому було проведено математичне моделювання задачі на паралельних комп'ютерах різної архітектури, використовуючи створене алгоритмічно-програмне забезпечення для розв'язування часткової узагальненої АПВЗ. А саме задачу на власні значення для стрічкових додатно визначених матриць було розв'язано методом ітерацій на підпросторі [6, 7], використовуючи новий гібридний алгоритм та

паралельний алгоритм для комп’ютерів з новітніми процесорами Intel Xeon Phi [16 – 18]. Задача досліджувалася при різних вихідних даних. Результати досліджень показали суттєве підвищення ефективності чисельного дослідження композитних матеріалів (див. розділ 3).

2.5 Висновки до розділу

Подана формалізована модель задачі стійкості композитних матеріалів нескінченно довгих смуг. Описана побудова дискретної та різницевої моделей, створена розрахункова схема задачі стійкості композитного матеріалу.

3 АЛГОРИТМИ РОЗВ'ЯЗУВАННЯ ЗАДАЧІ

Розв'язування багатьох практичних задач методом скінченних елементів або скінченних різниць для диференціальних рівнянь (наприклад, дослідження стійкості конструкцій) зводиться до часткової узагальненої АПВЗ для стрічкових симетричних матриць великого розміру.

У цьому розділі побудовано і досліджено гібридний алгоритм методу ітерацій на підпросторі для розв'язування часткової узагальненої АПВЗ (кілька власних значень) стрічкових симетричних матриць, одна з яких додатно визначена, на гібридних комп'ютерах, що поєднують багатоядерні центральні процесори з розподіленою пам'яттю та масивно-паралельні графічні прискорювачі.

Запропоновано ефективний розподіл даних між обчислювальними пристроями гібридного комп'ютера. Проведено дослідження прискорення та ефективності створеного алгоритму.

Подано результати експериментального дослідження нового гібридного алгоритму при математичному моделюванні процесів в різних предметних областях, які зводяться до розв'язування часткової узагальненої АПВЗ.

Розробка гібридного алгоритму розв'язування часткової узагальненої АПВЗ, що пропонується, заснована на методі ітерацій на підпросторі.

3.1 Метод ітерацій на підпросторі

Розглянемо метод ітерацій на підпросторі для обчислення r мінімальних власних значень і відповідних їм власних векторів задачі

$$Ax = \lambda Bx, \tag{3.1}$$

де A, B – симетричні стрічкові матриці порядку n, матриця A – додатно визначена.

Цей метод полягає у побудові для задачі (3.1) послідовності підпросторів E_t $(t = 1,2,\ldots)$, яка збігається до підпростору E_∞, що містить шукані власні вектори [1].

На t-ій ітерації обчислюється ортогональний базис підпростору E_t і, якщо досягнута необхідна точність наближеного розв'язку, визначаються шукані власні пари.

Таким чином, реалізація методу ітерацій на підпросторі задачі (3.1) зводиться

до виконання для $t = 1,2,\ldots$ таких кроків:

- знаходження розв'язку СЛАР

$$AX_t = Y_{t-1}; \tag{3.2}$$

- обчислення проекції матриці A на підпростір E_t

$$A_t = X_t^T Y_{t-1} \equiv X_t^T A X_t; \tag{3.3}$$

- обчислення прямокутної матриці

$$W_t = BX_t; \tag{3.4}$$

- обчислення проекції матриці B на підпростір E_t

$$B_t = X_t^T W_t \equiv X_t^T B X_t; \tag{3.5}$$

- розв'язування повної проблеми власних значень для проекцій

$$A_t Z_t = B_t Z_t \Lambda_t, \text{ де } \Lambda_t = diag(\lambda_i); \tag{3.6}$$

- обчислення наближення

$$Y_t = W_t Z_t. \tag{3.7}$$

Якщо після t ітерацій виконуються умови закінчення ітераційного процесу, наприклад, $\left|\frac{\lambda_i^{(t)} - \lambda_i^{(t-1)}}{\lambda_i^{(t)}}\right| \leq \varepsilon$, то проводиться додаткова ітерація і наближеними розв'язками задачі (3.1) приймаються власні значення – $\lambda_i^* = \lambda_i^{(c+1)}$, де $i = 1,2,\ldots,r$, та власні вектори – перші r стовпчиків матриці $X^* = X_{c+1}Z_{c+1}$. Мається на увазі, що власні значення упорядковані за зростанням, тобто $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_r \leq \ldots$

З [1] відомо, що ітераційний процес збігається лінійно, причому швидкість збіжності λ_i визначається відношенням λ_q/λ_1, де q – розмір підпростору E_q, що ітерується. В послідовних реалізаціях алгоритму рекомендується вибирати $q = \min(2r, r+8)$.

Аналізуючи схему реалізації методу ітерацій на підпросторі, можна відзначити, що найбільш витратною математичною операцією щодо використання обчислювальних ресурсів та комп'ютерного часу є розв'язування СЛАР одним із алгоритмів методу Холецького, в основі якого покладено наступну теорему [3]. Якщо A – симетрична додатно визначена матриця, то існує дійсна невироджена

нижня трикутна матриця L така, що $LL^T = A$.

Оскільки в обчислювальній схемі (3.2)–(3.7) на кожній ітерації виконується розв'язування СЛАР (3.2) з однією і тією ж матрицею A, то LL^T-розвинення цієї матриці доцільно виконати до початку ітераційного процесу. Тоді на кожній ітерації розв'язується задача (3.2) із факторизованою матрицею, що значно зменшує час виконання обчислень.

Отримавши в комп'ютері розв'язок часткової узагальненої АПВЗ симетричних стрічкових матриць, необхідно дослідити, що знайдено всі потрібні власні значення та відповідні власні вектори. Адже при обчисленні k мінімальних власних значень може виникнути ситуація, коли замість одного з них обчислено $(k+1)$ -е власне значення.

Зокрема, подібні ситуації можуть виникати у випадку, коли початковий підпростір E_0 виявиться B-ортогональним до принаймні одного з шуканих власних векторів. До того ж більш складно забезпечити апріорну B-ортогональність початкового підпростору до всіх шуканих векторів, розв'язуючи задачу на MIMD-комп'ютері або на гібридному комп'ютері. Адже при реалізації паралельного алгоритму методу ітерацій на підпросторі матриця векторів, на які натягнуто початковий підпростір, формується з розподілених її частин між паралельними обчислювальними пристроями.

Виявити ситуацію, коли через ортогональність до початкового підпростору не знайдено одну (або більше) з шуканих власних пар, можна за властивістю послідовності Штурма для задачі (3.1).

Ця властивість базується на тому факті, що за умови додатної визначеності матриці B кількості від'ємних, нульових і додатних власних значень задачі (3.1) і матриці A співпадають. А кількості від'ємних і додатних власних значень невиродженої симетричної матриці дорівнюють відповідно кількостям від'ємних і додатних елементів діагональної матриці D з LDL^{T}-розвинення цієї матриці. Отже, для аналізу достовірності комп'ютерного розв'язку задачі (3.1) можна використати алгоритм LDL^T-розвинення стрічкових симетричних додатно визначених матриць [3, 5].

Якщо матриця B задачі (3.1) додатно напіввизначена, а матриця A додатно визначена, то те ж саме справедливо стосовно першої матриці задачі $Bx = \mu Ax$. Зауважимо, що власні значення цієї задачі та задачі (3.1) зв'язані співвідношенням $\lambda_i \mu_i = 1$ за умов $\lambda_i \neq 0,\ \mu_i \neq 0$. Виконавши LDL^{T}-розвинення матриці $A - \mu B$ і підрахувавши кількість від'ємних елементів діагональної матриці D, можна визначити кількість власних значень задачі (3.1) менших за μ. Через те, що матриця $A - \mu B$ є знаконевизначеною, для забезпечення стійкості при LDL^{T}-розвиненні зсув μ необхідно вибирати більшим за максимальне з власних значень, які відшукуються, враховуючи кратність або патологічну близькість в межах точності обчислень, наприклад, $\mu = 1{,}01\lambda_r$.

Для оцінки відносної обчислювальної похибки власних значень має місце наступна нерівність [2, 6]:

$$\min_j \left| \frac{\lambda_j - \lambda_i^*}{\lambda_j} \right| \leq \sqrt{\frac{\bar{r}_i^T B \bar{r}_i}{x_i^{*T} B x_i^*}}, \qquad i = 1,\ 2,\ \ldots,\ r,$$

де λ_i^*, x_i^*– обчислена наближена власна пара

$\bar{r}_i = A^{-1} r_i = x_i^* - \lambda_i^* A^{-1} B x_i^*$.

3.2 Гібридний алгоритм методу ітерацій на підпросторі

Для розв'язування часткової узагальненої проблеми власних значень (знаходження кількох мінімальних власних значень та відповідних їм власних векторів) задачі (3.1) з дійсними стрічковими симетричними матрицями, при цьому матриця A є додатно визначеною, пропонується гібридний алгоритм.

Якщо матриці задачі (3.1) є дійсними симетричними, то власні значення є дійсними числами, які можуть бути впорядковані, наприклад, за зростанням, і таким чином перенумеровані. Власні вектори задачі з дійсними симетричними матрицями є також дійсні.

3.2.1 Розподіл даних між обчислювальними пристроями гібридного комп'ютера

Розглядаємо гібридний комп'ютер, архітектура якого складається з багатоядерного комп'ютера MIMD-архітектури із топологією комунікаційних

зв’язків між процесорами – «гіперкуб», створеною за допомогою системи MPI, та декількох графічних SIMT-процесорів (GPU). Тобто маємо гібридний комп’ютер з pCPU та pGPU, p – загальна кількість процесів, що виконуються на центральному процесорі (CPU).

Для розв’язування задачі (3.1) на гібридному комп’ютері розділимо матрицю A на квадратні блоки $A_{I,J}$ порядку s. Елементи головної діагоналі та нижнього або верхнього трикутника (в залежності від використаних алгоритмів) ненульових блоків цієї стрічкової симетричної матриці розподілимо між процесами CPU у відповідності з одномірною блочно-циклічною схемою .

За цією схемою блок $A_{I,J}$ зберігається в процесі CPU з логічним номером $(I + l)\mathrm{mod}p$ (результат операції $k\mathrm{mod}j$ – залишок від ділення k на j, $-1 \le l \le p-2$ – зсув, зазвичай $l = -1$).

У результаті виконання LL^T-розвинення матриці A блоки нижньої трикутної матриці L або верхньої трикутної матриці L^T будуть розподілені аналогічно. Така ж блочно-циклічна схема розподілу використовується для елементів матриці B та прямокутних матриць ітерованих векторів X_t, Y_t, W_t. При цьому достатньо розподіляти та зберігати лише ненульові елементи матриці B у такій послідовності: піддіагональні, діагональні та надіагональні. Це значно спрощує алгоритм перемноження такої матриці на прямокутну матрицю, несуттєво збільшуючи загальний обсяг даних.

3.2.2 Реалізація гібридного алгоритму методу ітерацій на підпросторі

На основі проведеного аналізу методу ітерацій на підпросторі та з метою ефективного використання архітектурних і технологічних особливостей гібридного комп’ютера етапи розв’язування задачі (3.1) розділимо на чотири підзадачі різної обчислювальної складності.

Підзадача1. Формування розподіленої між задіяними процесами на CPU матриці Y_0 початкових векторів, що ітеруються, таких, щоб матриця B_t була додатно визначеною. Оскільки підматриці E_t ітеровних векторів розподілені між процесами блоками рядків, то цю операцію можна виконати в кожному з них без обмінів.

Підзадача 2. LL^T-факторизація стрічкової симетричної додатно визначеної матриці A, використовуючи, наприклад, гібридний блочний алгоритм.

Підзадача 3. виконання ітераційного процесу (3.2) – (3.7), за яким на кожній ітерації $(t = 1,2,\ldots)$ обчислення виконуються на процесах CPU за наступною схемою:

Крок 1. Розв'язування гібридними алгоритмами системи лінійних рівнянь (3.2) з трикутними матрицями, використовуючи одержане на попередньому кроці LL^T-розвинення матриці A;

Крок 2.Обчислення прямокутної матриці $W_t = BX_t$, тобто добутку прямокутної матриці на стрічкову матрицю (3.4), що виконується в такому порядку:

2.1 пересилка в кожний процес CPU з процесу, де вони постійно розташовані, елементів (рядків) прямокутної матриці, що використовуються для обчислення чергових kp рядків матриці W_t;

2.2 обчислення процесами, на відповідних процесорах GPU, часткових сум для елементів чергових kp рядків матриці W_t;

2.3 мультиобмін чергових kp рядків матриці W_t, тобто мультизбирання рядків часткових сум елементів, яке можна поєднати з їх розподілом між процесами CPU у відповідністю зі схемою зберігання.

Крок 3: обчислення добутків (3.3) та (3.5) прямокутних матриць для формування проекцій матриць A та B на підпростір. Виконуються кожним процесом на відповідних GPU, після чого здійснюється збір результуючих матриць проекцій. Причому обчислення добутків прямокутних матриць можна виконувати асинхронно з іншими обчислювальними операціями або обмінами на CPU;

Крок 4: Розв'язування повної узагальненої АПВЗ (3.6) методом Якобі, враховуючи порівняно невеликий порядок матриць проекцій підзадачі, виконуються процесами на CPU без обмінів даними [2].

Крок 5: Перевірка умов закінчення ітераційного процесу в кожному процесі CPU.

Крок 6: Обчислення (3.7) нової матриці ітерованих векторів Y_t (або матриці наближених власних векторів X*) виконується на процесорах GPU у відповідності з розподілом даних (обчислюється підматриця матриці Y_t або X*), причому немає необхідності в обмінах даними між процесорними пристроями.

Підзадача 4. Аналіз одержаних результатів.

Результатом роботи гібридного алгоритму є:

- обчислені власні значення λ_i, розташовані у кожному процесі CPU в порядку зростання;
- матриця власних векторів X, відповідних обчисленим власним значенням (елементи цієї матриці розподілені між процесами відповідно до одномірної блочно-циклічної схеми, яка співпадає зі схемою розподілу елементів матриць A та B).

Слід відзначити, що обсяг обчислень для розв'язування СЛАР (3.2) на порядок або більше перевищує обсяг обчислень у інших підзадачах. Причому значна частина часу витрачається на LL^T-розвинення матриці. Отже, ефективність гібридного алгоритму методу ітерацій на підпросторі визначається, перш за все, ефективним виконанням гібридного алгоритму розв'язування СЛАР.

Розглянемо реалізацію гібридного блочного алгоритму розв'язування СЛАР із стрічковою додатно визначеною матрицею на комп'ютері з декількома процесами CPU та відповідними процесорами GPU.

Як відомо, алгоритм розв'язування СЛАР

$$Ax = b, \tag{3.8}$$

реалізується за такою обчислювальною схемою:

- LL^T-розвинення (факторизація) матриці A:

$$A = L \times L^T; \tag{3.9}$$

– розв’язування двох СЛАР з трикутними матрицями

$$Ly = b, \tag{3.10}$$

$$L^T x = y. \tag{3.11}$$

Для реалізації гібридного алгоритму факторизації стрічкової додатно визначеної матриці (3.9) використаємо блочно-циклічний розподіл матриць та векторів між процесами CPU.

Розділимо стрічкову матрицю A, порядок якої n та напівширина стрічки m, на блоки порядку s. Тобто блок A_{ij} є підматриця матриці A, що знаходиться на перетині рядків $\overline{(i-1)s+1, is}$ та стовпчиків $\overline{j(s-1)+1, js}$.

Для зручності, в подальшому будемо вважати, що порядок матриці n та її напівширина стрічки m кратні числу s. Тоді позначимо $q = \frac{n}{s}, l = \frac{m}{s}$.

Оскільки матриця A симетрична, то для її трикутного розвинення (3.9) достатньо розглядати тільки нижній трикутник одержаної блочної матриці:

$$\begin{pmatrix} A_{11} & 0 & 0 & & & 0 \\ A_{21} & A_{22} & 0 & & & 0 \\ A_{31} & A_{32} & \ddots & & & 0 \\ \vdots & \vdots & & & & \\ A_{(l+1)1} & A_{(l+1)2} & & & & \\ 0 & A_{(l+2)2} & & & & \\ 0 & 0 & & & & \\ \vdots & \vdots & & \dots & A_{(q-1)(q-1)} & 0 \\ 0 & 0 & & \dots & A_{q(q-1)} & A_{qq} \end{pmatrix}.$$

Схематично блочно-циклічний розподіл у стрічці матриці показано на рисунку 3.1 для випадку використання трьох процесів CPU ($p = 3$).

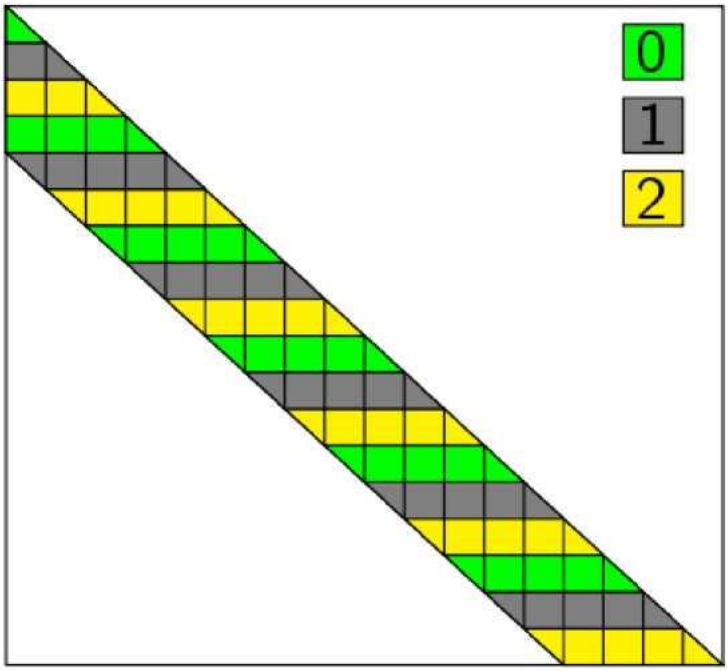

Рисунок 3.1 - Блочно-циклічний розподіл матриці

За таким розподілом матриці між процесами математичні операції з окремими елементами замінюються математичними операціями з блоками розміром $s \times s$.

Реалізація гібридного блочного алгоритму LL^T-розвинення стрічкової матриці при використанні pCPU та відповідних pGPU складається з q кроків. На i-му кроці обчислюються блоки матриці L в i-му стовпчику. Будемо називати процеси CPU та відповідні до них процесори GPU головними (провідними) на тих кроках обчислень, де A_{ii} та L_{ii} знаходяться в їх пам'яті.

На i-му кроці алгоритму факторизації необхідно оперувати лише з тими блоками, які знаходяться в i-му стовпчику та в l стовпчиках, наступних за ним. З блочного представлення матриці A видно, що в кожному стовпчику знаходиться не більше $l + 1$ блоків. Отже, на кожному кроці алгоритму достатньо зберігати в пам'яті $(l + 1)^2$ блоків.

Перед початком виконання факторизації стрічкової додатно визначеної матриці (3.9) необхідно здійснити копіювання блоків матриці A, які знаходяться в перших $l + 1$ стовпчиках на CPU, в пам'ять відповідних GPU: блок A_{ij} копіюється в GPU за номером $(i - 1) \bmod p$.

Отже, за гібридним алгоритмом факторизації матриці A для кожного $i =$

$1,2,\ldots,q$ виконуються наступні кроки:

а) LL^T-факторизація діагонального блоку A_{ii} на провідному CPU

$$A_{ii} \to L_{ii}L_{ii}^T;$$

б) розсилка провідним CPU блоку L_{ii} усім GPU, що використовуються;

в) обчислення на відповідних GPU блоків L_{ji}, де $j = \overline{i+1,\min(i+l,q)}$, за формулою $L_{ji} = A_{ji}(L_{ii}^T)^{-1}$;

г) асинхронне обчислення блоків A_{jt} на відповідних GPU, де $j = \overline{i+1,\min(i+l,q)}$, $t = \overline{j,\min(i+l,q)}$, $t-j \equiv 0(\mathrm{mod}\, p)$, за формулою $A_{jt} \leftarrow A_{jt} - L_{ji}(L_{ti})^T$;

д) кожний CPU виконує розсилку обчислених на кроці 3 блоків усім іншим процесам;

е) оновлення блоків A_{jt} на відповідних GPU, де $j = \overline{i+1,\min(i+l,q)}$, $t = \overline{j,\min(i+l,q())}$ та $(t-j)$ не ділиться на p;

ж) асинхронне копіювання, у разі $i+l+2 < q$, всіх блоків матриці, які знаходяться у стовпчику $i+l+2$, в пам'ять відповідних GPU;

з) синхронне обчислення в CPU з логічним номером $i\mathrm{mod}p$ блоку $A_{i+1,i+1}$;

и) копіювання блоку $A_{i+1,i+1}$ в пам'ять відповідного CPU.

Далі розглянемо реалізацію ***гібридного алгоритму розв'язування СЛАР з трикутними матрицями*** за формулами (3.10), (3.11), він, як і факторизація матриці, виконуються з використанням pCPU та відповідних pGPU.

Розв'язування СЛАР (3.10) з нижньою трикутною матрицею

Для кожного $i = \overline{1,\ q}$ виконати наступні кроки:

а) провідний процесор GPU виконує розсилку діагонального блоку L_{ii} усім іншим GPU, що задіяні в обчисленнях;

б) всі GPU виконують розв'язування СЛАР з нижньою трикутною матрицею L_{ii} та правою частиною b_k, де $k = \overline{1,\ q}$, за формулою

$$L_{ii}y_k = b_k;$$

в) кожний CPU виконує розсилку блоків L_{ij}, де $j = \overline{i+1,\min(i+l,q())}$, всім

іншим CPU;

г) відповідні GPU виконують оновлення вектора b_k, де $k = \overline{1,\ q}$, за формулою

$$b_k \leftarrow b_k - L_{ji} y_k.$$

Після виконання обчислень у кожному CPU, при використанні їхніх GPU, елементи вектора *y* записуються замість відповідних елементів вектора *b*.

Розв'язування СЛАР (3.11) з верхньою трикутною матрицею

Для кожного i від q до 1 виконати наступні кроки:

а) GPU, відповідний провідному CPU, виконує розсилку діагонального блоку L_{ii} всім іншим задіяним GPU;

б) всі GPU виконують розв'язування СЛАР з верхньою трикутною матрицею $(L_{ii})^T x_k = y_k, \quad k = \overline{1,\ q}$;

в) кожний CPU виконує розсилку одержаного розв'язку x_k всім іншим GPU;

г) на відповідних GPU виконується оновлення векторів y_j,

де $j = \overline{\min(i-l,1), i+1}$, за формулою

$$y_j \leftarrow y_j - L_{ij}^T x_k, \quad k = \overline{1,\ q}.$$

При цьому обчислення на кроці 4 проводяться на всіх GPU, а кожен вектор y_j буде змінюватися лише на GPU з логічним номером $k - 1 \bmod p$.

3.2.3 Прискорення та ефективність гібридного алгоритму методу ітерацій на підпросторі

Дослідження ефективності запропонованого гібридного алгоритму базується на методологіях, які описано в роботах [2, 6].

Нагадаємо, що ефективність паралельних алгоритмів характеризується коефіцієнтами прискорення $S_p = T_1/T_p$ та коефіцієнтами ефективності $E_p = S_p/p$.

Для гібридних комп'ютерів позначимо: T_1 – час розв'язування задачі на архітектурі $1\text{CPU} + 1\text{GPU}$, T_p – час розв'язування цієї ж задачі на архітектурі $p\text{CPU} + p\text{GPU}$.

Тоді час розв'язування задачі за розробленим гібридним алгоритмом можна

обчислити за формулами:

$$T_1 = O_1 t_C + O_{1G}\, t_G/n_o + O_{oG} t_{oG} + O_{cG} t_{cG},$$
$$T_p = O_p t_C + O_{pG}\, t_G/n_o + O_o t_o + O_{oG} t_{oG} + O_c t_c + O_{cG} t_{cG}. \quad (3.12)$$

У формулах (3.12) маємо:

- O_1, O_p – кількість операцій, що виконуються на CPU, O_{1G}, O_{pG} – кількість операцій, що виконуються на GPU, відповідно послідовною та паралельною версією алгоритму;
- t_C, t_G – середній час виконання однієї арифметичної операції з плаваючою комою відповідно на CPU та GPU;
- n_o – кількість операцій, які можуть бути одночасно виконані на GPU;
- t_o, t_{oG} – час, необхідний для обміну одним машинним словом між процесами CPU або між CPU та GPU відповідно;
- O_o, O_{oG} – обсяг обмінів (кількість машинних слів), що виконуються одним CPU та GPU відповідно;
- t_c, t_{cG} – час для синхронізації двох CPU або CPU та його GPU відповідно;
- O_c, O_{cG} – кількість синхронізацій, що виконуються одним CPU або GPU відповідно.

Якщо задача розв'язується на комп'ютері гібридної архітектури, то у формулах (3.12) необхідно враховувати синхронність або асинхронність роботи CPU та GPU. У разі асинхронної роботи обчислення суми потрібно замінять визначенням максимального значення.

При виконанні дослідження та розв'язування часткової узагальненої АПВЗ (3.1) для симетричних матриць, частина етапів (формування матриці початкових ітерованих векторів, LL^T-розвинення матриці A) має фіксовану кількість арифметичних операцій, а для ітераційного процесу (3.2) – (3.7) кількість арифметичних операцій є пропорційною кількості виконаних ітерацій.

Кількість ітерацій, яка необхідна для знаходження r мінімальних власних значень, як правило, є величина $O(r)$, причому вона тим менше, чим більший розмір q ітерованого підпростору.

Таким чином, $T_p = T_p^{(F)} + c_I T_p^{(1I)}$, а коефіцієнт прискорення запропонованого гібридного алгоритму можна представити у вигляді

$$S_p = \frac{T_1}{T_p} = \frac{T_p^{(F)}}{T_p} S_p^{(F)} + \frac{c_I T_p^{(1I)}}{T_p} S_p^{(It)} \tag{3.13}$$

У формулі (3.13) маємо c_I – кількість ітерацій; $T_p^{(F)}$, $T_p^{(1I)}$– відповідно час розв'язування підзадач (**1**), (**2**) з фіксованою кількістю арифметичних операцій та час виконання однієї ітерації в підзадачі (**3**) за формулами (3.2)–(3.7) на гібридній архітектурі $p\text{CPU} + p\text{GPU}$; $S_p^{(F)}$, $S_p^{(1I)}$ – коефіцієнти прискорення алгоритмів, що використовуються для розв'язування цих підзадач.

Далі введемо для архітектури $p\text{CPU} + p\text{GPU}$ такі позначення: $T_p^{(LLT)}$ – час виконання LL^T-розвинення матриці A; $T_p^{(SS)}(k)$ – час розв'язування СЛАР вигляду (3.2) з k правими частинами (використовуючи обчислене раніше LL^T-розвинення матриці); $T_p^{(Fo)}$, $T_p^{(Io)}(k)$ – відповідно час виконання інших операцій для реалізації розв'язування підзадач з фіксованою кількістю арифметичних операцій та час виконання однієї ітерації.

Тоді $T_p^{(F)}$, $T_p^{(1I)}$ можна представити формулами:

$$T_p^{(F)} = T_p^{(LLT)} + T_p^{(SS)}(q) + T_p^{(Fo)}, \quad T_p^{(1I)} = T_p^{(SS)}(q) + T_p^{(Io)}(q). \tag{3.14}$$

Слід зауважити, що тут і далі формули для всіх $T_1^{(*)}$ можна отримати, підставивши у формули для $T_p^{(*)}$ значення $p = 1$. Для їх обчислення справедливі формули (3.12).

Тут і далі замість * підставляємо позначення відповідної підзадачі або математичної операції.

Коефіцієнт прискорення гібридного алгоритму методу ітерацій на підпросторі для архітектури $p\text{CPU} + p\text{GPU}$, базуючись на (3.13), можна обчислити за формулою

$$S_p = \frac{T_p^{(F)}}{T_p}\left(\alpha_p^{(LLT)} S_p^{(LLT)} + \alpha_p^{(SS)}(q) S_p^{(SS)}(q) + \alpha_p^{(Fo)} S_p^{(Fo)}\right) +$$

$$+\frac{c_I T_p^{(1I)}}{T_p}\left(\beta_p^{(SS)}(q)S_p^{(SS)}(q)+\beta_p^{(Io)}(q)S_p^{(Io)}(q)\right), \quad (3.15)$$

де $\alpha_p^{(*)}=T_p^{(*)}/T_p^{(F)}$, $\beta_p^{(*)}=T_p^{(*)}/T_p^{(1I)}$, $S_p^{(*)}$ – коефіцієнт прискорення гібридного алгоритму розв'язування відповідної підзадачі (виконання математичних операцій), а верхні індекси і параметри співпадають з відповідними індексами і параметрами в (3.14) для $T_p^{(*)}$.

Ми розглядаємо гібридний алгоритм розв'язування часткової проблеми власних значень (3.1) для стрічкових матриць, тому введемо додаткові позначення: $m_A,\ m_B$ – напівширина стрічки матриць A та B відповідно. Також будемо вважати, що $n=O(10^k m_A),\ \ k>1,\ m_A\geq m_B,\ m_A>>q,\ \eta_B$ – середня кількість ненульових елементів в одному рядку матриці B (зазвичай, $\eta_B << m_B$).

Якщо кількість використаних процесів $p>1$, то обміни між процесами на CPU – це операції мультирозсилки та мультизбирання, для реалізації яких використовується алгоритм «дерево».

Час обміну масивом з d подвійних слів між двома CPU складає $t_C^{(E)}(d)=t_c+dt_o$, а між CPU та відповідним GPU маємо $t_G^{(E)}(d)=t_{cG}+dt_{oG}$.

Для випадку реалізації гібридного блочного алгоритму розв'язування СЛАР із симетричною стрічковою матрицею (нижньою або верхньою) можна сформулювати лему (нехтуючи величинами невеликих порядків).

Лема 3.1. Для розв'язування СЛАР із стрічковою симетричною додатно визначеною матрицею A розміру $n>>q>r,\ m_A>sp$ гібридним алгоритмом на основі LL^T–розвинення на комунікаційній мережі «гіперкуб» при використанні pCPUта pGPUсправедливі такі оцінки часових характеристик:

$$T_p^{(LLT)}\approx\frac{n}{p}\left(\frac{s^2}{3}pt_C+\max\left\{m_A^2\frac{t_G}{n_o},\ t_C^{(E)}(sm_A)\frac{p}{s}\log_2 p\right\}+2psm_A\frac{t_G}{n_o}+t_G^{(E)}(sm_A)\frac{p}{s}\right), \quad (3.16)$$

$$T_p^{(SS)}(k)=\frac{n}{p}\left(kt_C p\log_2 p+(4m_A+2ps)k\frac{t_G}{n_o}+(t_C^{(E)}(sk)\log_2 p+2t_G^{(E)}(sk))\frac{2p}{s}\right). \quad (3.17)$$

З (3.16), (3.17), $t_G^{(m)}(s)=\frac{m_A^2 s}{p\,log_2\,p}\times\frac{t_G}{n_o}$, враховуючи можливість одночасного виконання низки операцій обчислень і обмінів на CPU і GPU або різними потоками на GPU, а також, нехтуючи доданками меншого порядку, одержуємо такі оцінки

коефіцієнтів прискорення для гібридних алгоритмів LL^T-розвинення стрічкової симетричної матриці та розв'язування СЛАР з факторизованою матрицею виду (3.2), на архітектурі $p\text{CPU} + p\text{GPU}$:

$$S_p^{(LLT)} \approx p\left(1 - \frac{2s(p-1)}{m_A+2ps}\right), \qquad t_G^{(m)}(s) \geq t_C^{(E)}(sm_A);$$

$$S_p^{(LLT)} \approx p\left(\frac{2ps}{m_A+2s} + \frac{p\log_2 p}{m_A s(m_A+2s)} \times \frac{n_o t_C^{(E)}(sm_A)}{t_G}\right)^{-1}, \qquad (3.18)$$

$$t_G^{(m)}(s) < t_C^{(E)}(sm_A), \quad t_G^{(m)}(s) = \frac{m_A^2 s}{p\log_2 p}\frac{t_G}{n_o};$$

$$S_p^{(SS)}(k) \approx p\left(1 + \frac{(p-1)s}{2m_A+s} + \frac{p\log_2 p}{ks(2m_A+s)} \times \frac{n_o t_C^{(E)}(ks)}{t_G}\right)^{-1}.$$

При визначенні оцінок інших часових характеристик – $T_p^{(Fo)}$та $T_p^{(Io)}(k)$ необхідно враховувати, що в загальному випадку найбільша кількість арифметичних операцій виконується при обчисленні добутку розрідженої матриці B на $(n \times k)$ - матрицю та при обчисленні матриць або векторів, елементами яких є скалярні добутки n-мірних векторів. Кількість інших арифметичних операцій, які виконуються одним з p GPU, значно менше і нею можна знехтувати.

Тоді справедливе наступне твердження.

Лема 3.2. Для гібридного алгоритму методу ітерацій на підпросторі розв'язування АПВЗ для стрічкових симетричних матриць, за умов $n >> q > r$, $m_A \geq m_B$, $1 \leq \eta_B << m_B$, час виконання обчислень з фіксованою кількістю арифметичних операцій – $T_p^{(Fo)}$ та час виконання однієї ітерації – $T_p^{(Io)}(k)$ оцінюються за формулами:

$$T_p^{(Fo)} = \frac{n}{p}\left(\frac{n+2(n+2)p\log_2 p}{n} q t_C + (2\eta_B + 6) q \frac{t_G}{n_o}\right) + \\ + (2t_C^{(E)}(sq)\log_2 p + 2t_G^{(E)}(sq))\frac{n}{s}, \qquad (3.19)$$

$$T_p^{(Io)}(k) = \frac{n}{p}\left(\frac{(n+2k)\log_2 p + O(k^2)}{n} p k t_C + (2\eta_B + 4k) k \frac{t_G}{n_o}\right) + \\ + (t_C^{(E)}(sk)\log_2 p + 2t_G^{(E)}(sk))\frac{n}{s}. \qquad (3.20)$$

Оскільки в (3.19) значення $T_p^{(Fo)}$ суттєво менше значень $T_p^{(LLT)}$ та $T_p^{(SS)}(q)$ з (3.14), то ним можна знехтувати, поклавши в (3.15) $\alpha_p^{(Fo)} \cong 0$.

Вплив $T_p^{(Io)}(g)$ на час виконання алгоритму залежить від співвідношень m_A, η_B та q. Якщо q суттєво більше за η_B, то в (3.20) можна знехтувати доданком, що містить η_B. Якщо $m_A >> \eta_B$ та $m_A >> q$, то в (3.14) можна знехтувати $T_p^{(Io)}(g)$.

Враховуючи часові характеристики з (3.20), одержуємо наступну оцінку

$$S_p^{(Io)}(k) \approx p\left(1 + \frac{p\log_2 p}{2k(\eta_B+2k)} \times \frac{n_o}{t_G}(kt_C + t_C^{(E)}(sk))\right)^{-1}. \qquad (3.21)$$

Тепер можна оцінити прискорення гібридного алгоритму в цілому наступною теоремою.

Теорема 3.1. Для гібридного алгоритму методу ітерацій на підпросторі розв'язування часткової узагальненої АПВЗ стрічкових матриць, за умов $n >> q$, $m_A \geq m_B$, $m_A >> q$, $m_A > sp$, $\eta_B << m_B$, справедливі оцінки коефіцієнтів прискорення:

$$S_p \approx \frac{\gamma_p}{\gamma_p + c_I}(\alpha_p^{(LLT)} S_p^{(LLT)} + \alpha_p^{(SS)}(q) S_p^{(SS)}(q)) + \\ + \frac{c_I}{\gamma_p + c_I}(\beta_p^{(SS)}(q) S_p^{(SS)}(q) + \beta_p^{(Io)}(q) S_p^{(Io)}(q)), \qquad (3.22)$$

де $\gamma_p = (T_p^{(LLT)} + T_p^{(SS)}(q))/T_p^{(1I)}$, а числові значення коефіцієнтів $\alpha_p^{(*)} = T_p^{(*)}/T_p^{(F)}$ та $\beta_p^{(*)} = T_p^{(*)}/T_p^{(1I)}$ отримуються з формул (3.14), (3.16), (3.19), (3.20).

В одержаних виразах малі складові щодо інших доданків відкидаються. Коефіцієнти прискорення відповідної підзадачі (з фіксованою кількістю операцій) – $S_p^{(*)}$ представлені формулами (3.18), (3.21), а верхні індекси і параметри співпадають з відповідними індексами і параметрами в (3.13) для часів $T_p^{(*)}$.

Аналіз отриманих оцінок (3.16) – (3.22) свідчить, що ефективність запропонованого алгоритму значною мірою визначається ефективністю гібридних блочних алгоритмів LL^T-розвинення та розв'язування СЛАР з нижньою і верхньою

трикутними матрицями.

Отже для оцінки коефіцієнта прискорення справедлива наступна теорема.

Теорема 3.2. Для гібридного алгоритму методу ітерацій на підпросторі, за умов $n >> q, \quad m_A \geq m_B, \quad m_A >> q, \quad m_A > sp, \quad \eta_B << m_B$та $t_G^{(m)}(s) \geq t_C^{(E)}(sm_A),$ для коефіцієнта прискорення справедлива наступна оцінка

$$S_p \approx \frac{1}{T_p^{(SLAE)} + c_I T_p^{(1I)}} \left(\frac{T_p^{(SLAE)} T_1^{(SLAE)} + c_I T_p^{(F)} T_1^{(1I)}}{T_p^{(F)}} \right),$$

де $T_p^{(SLAE)} = T_p^{(LLT)} + T_p^{(SS)}, \quad T_p^{(F)} = T_p^{(LLT)} + T_p^{(SS)} + T_p^{(Fo)}, \quad T_p^{(1I)} = T_p^{(SS)} + T_p^{(Io)}.$

$$T_p^{(SLAE)} = \frac{n}{p} \left(\frac{t_G m_A (m_A + 2ps + 4k) + 2t_G psk + pn_o((4pk + m_A) t_G^{(E)} + s^2 t_C)}{n_o} \right),$$

$$T_p^{(F)} = \frac{n}{p} \left(\frac{s^2}{3} p t_C + \frac{t_G (m_A^2 + 2psm_A + 4km_A)}{n_o} + t_G^{(E)} p(2nq + m_A) + 2q(t_C + t_C^{(E)} n) \right),$$

$$T_p^{(1I)} = \frac{n}{p} \left(\frac{pk \log_2 p (2nt_C + kt_C + n^2 t_C^{(E)})}{n} + \frac{t_G (2k + 2m_A + ps) 2k + 2n_o pk t_G^{(E)} n}{n_o} \right).$$

Доведення теореми 3.2. Для визначення коефіцієнта прискорення гібридного алгоритму методу ітерацій на підпросторі використаємо твердження (3.22) та проведемо деякі спрощення виразів.

Обчислимо відповідні вирази для $\frac{\gamma_p}{\gamma_p + c_I}$ та $\frac{c_I}{\gamma_p + c_I}$, попередньо обчисливши $\gamma_p + c_I$ за формулою:

$$\gamma_p + c_I = \frac{T_p^{(LLT)} + T_p^{(SS)}(q)}{T_p^{(1I)}} + c_I = \frac{T_p^{(LLT)} + T_p^{(SS)}(q) + c_I T_p^{(1I)}}{T_p^{(1I)}}.$$

Тоді маємо:

$$\frac{\gamma_p}{\gamma_p + c_I} = \frac{T_p^{(LLT)} + T_p^{(SS)}(q)}{T_p^{(1I)}} \times \frac{T_p^{(1I)}}{T_p^{(LLT)} + T_p^{(SS)}(q) + c_I T_p^{(1I)}}$$

$$= \frac{T_p^{(LLT)} + T_p^{(SS)}(q)}{T_p^{(LLT)} + T_p^{(SS)}(q) + c_I T_p^{(1I)}},$$

$$\frac{c_I}{\gamma_p + c_I} = c_I \times \frac{T_p^{(1I)}}{T_p^{(LLT)} + T_p^{(SS)}(q) + c_I T_p^{(1I)}} = \frac{c_I T_p^{(1I)}}{T_p^{(LLT)} + T_p^{(SS)}(q) + c_I T_p^{(1I)}},$$

Далі обчислимо відповідні коефіцієнти $\alpha_p^{(*)}$, $\beta_p^{(*)}$та $S_p^{(*)}$, підставимо їх у формулу (3.22) та зведемо відповідні доданки до загального знаменника.

В результаті отримаємо наступний вираз:

$$S_p \approx \frac{T_p^{(LLT)} + T_p^{(SS)}(q)}{T_p^{(LLT)} + T_p^{(SS)}(q) + c_I T_p^{(1I)}} \times \left(\frac{T_p^{(LLT)}}{T_p^{(F)}} \frac{T_1^{(LLT)}}{T_p^{(LLT)}} + \frac{T_p^{(SS)}}{T_p^{(F)}} \frac{T_1^{(SS)}}{T_p^{(SS)}} \right) +$$

$$+ \frac{c_I T_p^{(1I)}}{T_p^{(LLT)} + T_p^{(SS)}(q) + c_I T_p^{(1I)}} \times \left(\frac{T_p^{(SS)}}{T_p^{(1I)}} \frac{T_1^{(SS)}}{T_p^{(SS)}} + \frac{T_p^{(Io)}}{T_p^{(1I)}} \frac{T_1^{(Io)}}{T_p^{(Io)}} \right).$$

Після деяких маніпуляцій по спрощенню попереднього виразу одержуємо

$$S_p \approx \frac{1}{T_p^{(LLT)} + T_p^{(SS)} + c_I T_p^{(1I)}} \left(\frac{(T_p^{(LLT)} + T_p^{(SS)})(T_1^{(LLT)} + T_1^{(SS)}) + c_I T_p^{(F)}(T_1^{(SS)} + T_1^{(Io)})}{T_p^{(F)}} \right).$$

Провівши заміну $T_p^{(SLAE)} = T_p^{(LLT)} + T_p^{(SS)}$ та використавши вирази з (3.14), отримуємо результуючу формулу

$$S_p \approx \frac{1}{T_p^{(SLAE)} + c_I T_p^{(1I)}} \left(\frac{T_p^{(SLAE)} T_1^{(SLAE)} + c_I T_p^{(F)} T_1^{(1I)}}{T_p^{(F)}} \right).$$

Для обчислення $T_p^{(F)} = T_p^{(LLT)} + T_p^{(SS)} + T_p^{(Fo)}$ використаємо формули (3.16), (3.17) та (3.19). Виконавши спрощення виразу та знехтувавши малими величинами, отримаємо наступне:

$$T_p^{(F)} = \frac{n}{p} \left(\frac{s^2}{3} p t_C + \frac{t_G(m_A^2 + 2psm_A + 4km_A)}{n_o} + t_G^{(E)} p(2nq + m_A) + 2q(t_C + t_C^{(E)} n) \right).$$

Для обчислення $T_p^{(1I)} = T_p^{(SS)} + T_p^{(Io)}$ використаємо формули (3.17) та (3.20). Після виконання деяких спрощень в одержаному виразі для $T_p^{(1I)}$ та знехтувавши в ньому малими величинами, отримаємо наступне:

$$T_p^{(1I)} = \frac{n}{p} \left(\frac{\mathrm{pk log_2 p}(2\mathrm{nt_C} + \mathrm{kt_C} + \mathrm{n^2 t_C^{(E)}})}{\mathrm{n}} + \frac{\mathrm{t_G}(2\mathrm{k} + 2\mathrm{m_A} + \mathrm{ps})2\mathrm{k} + 2\mathrm{n_o pkt_G^{(E)} n}}{\mathrm{n_o}} \right),$$

Теорему 3.2 доведено.

Зауваження. Якщо час, що витрачається на пересилку даних між CPU та GPU,

буде менший за час обчислень (обміни здебільшого виконуються на фоні обчислень), то ці алгоритми за рахунок використання асинхронності матимуть коефіцієнти ефективності близькі до одиниці.

На ефективність гібридного алгоритму, що розглядається, істотний вплив має також c_I – кількість виконаних ітерацій для досягнення необхідної точності. Зазвичай, при постійному r значення c_I тим менше, чим більший розмір ітерованого підпростору q. Як випливає з оцінок (3.17), (3.19), обсяг обчислень є пропорційний q. Однак в ряді випадків, при відносно малих значеннях q, виникає недостатня завантаженість GPU (тобто фактично зменшується значення n_o) при збільшенні кількості обмінів, що призводить до зниження реальної ефективності алгоритму. Збільшуючи розмір ітерованого підпростору (до моменту досягнення максимального значення n_o), можна скоротити час розв'язування задачі за рахунок зменшення кількості ітерацій.

3.2.4 Аналіз отриманих результатів

Апробація гібридного алгоритму методу ітерацій на підпросторі проводилась на вузлі СКІТ 4 (CPU – два вузла, у вузлі – два процесори Xeon 5606 по чотири ядра, 288 Gb RAM; 2 GPU Nvidia Tesla M2090).

Для дослідження прискорення гібридного алгоритму використовувалися розріджені матриці різної структури та різних порядків, в тому числі було використанано матриці з колекції Флоридського університету.

На графіках (рисунок 3.2) демонструється прискорення гібридного алгоритму методу ітерацій на підпросторі, яке одержано на вузлі СКІТ 4 при розв'язуванні АПВЗ для матриць:

- Dubcova3 – порядок матриці 146 689, кількість ненульових елементів 3 636 643;
- bmwcra_1 – порядок матриці 148 770, кількість ненульових елементів 10 641 602;
- Bone010 – порядок матриці 986 703, кількість ненульових елементів 47 851 783.

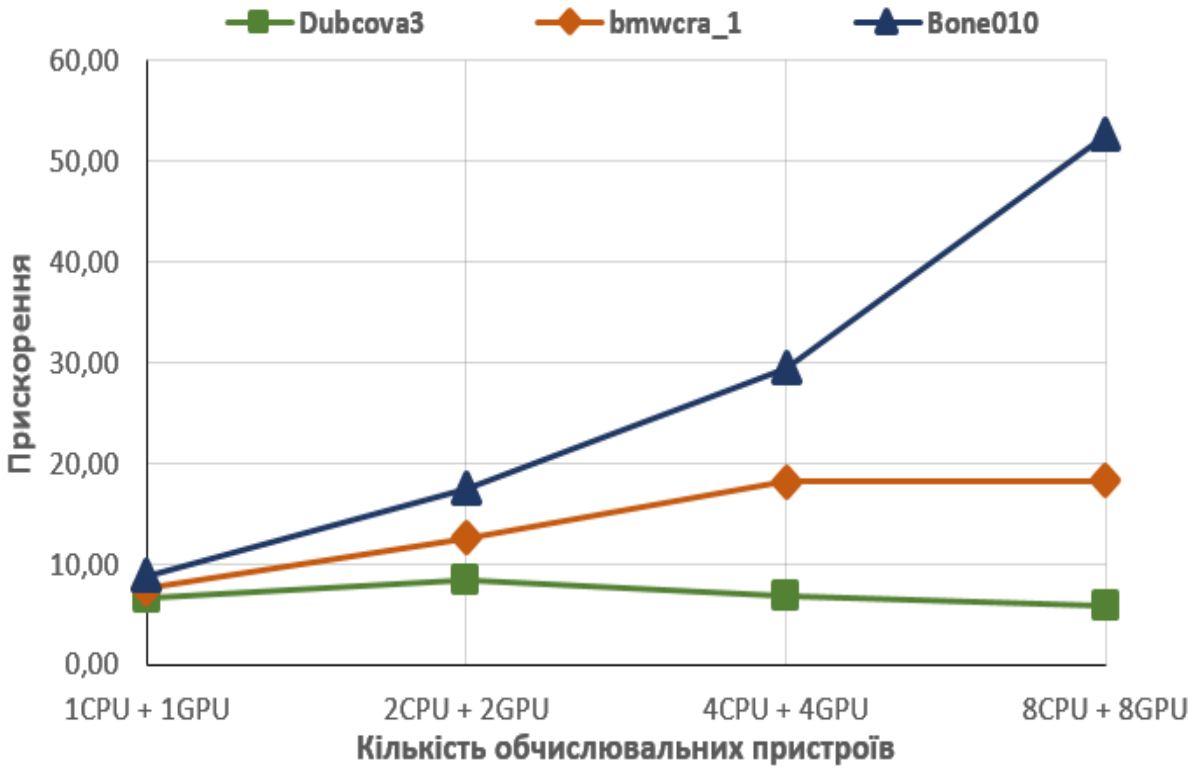

Рисунок 3.2 - Прискорення гібридного алгоритму методу ітерацій на підпросторі

З рисунку ми бачимо, що розроблений гібридний алгоритм добре масштабований, прискорення при збільшенні кількості процесів CPU та відповідних GPU значно зростає для всіх наведених задач.

Крім того, можна відзначити, що для матриць невеликого розміру (Dubcova3), при виході за межі обчислювального вузла (коли кількість використаних GPU більша двох) наступає насичення процесу обчислювальними ресурсами.

Водночас, для матриць великого порядку bmwcra_1 та Bone010 зі збільшенням кількості CPU та GPU прискорення зростає.

Проведено також дослідження залежності прискорення гібридного алгоритму від узгодженості розміру задачі та кількості використаних комп'ютерних ресурсів.

На рисунку 3.3 наведено залежність прискорення гібридного алгоритму від напівширини стрічки матриці та кількості використаних GPU, що отримано при розв'язуванні АПВЗ стрічкових симетричних матриць порядку 250 000 з різною напівшириною стрічки.

Матриці одтримано шляхом дискретизації методом скінченних елементів. Дослідження проведено на вузлі СКІТ 4.

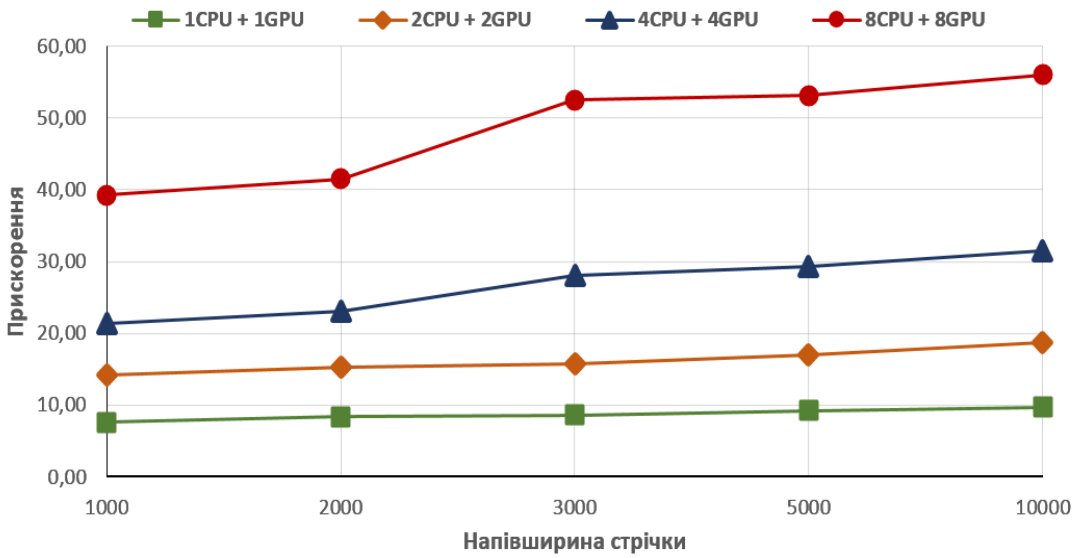

Рисунок 3.3 - Залежність прискорення гібридного алгоритму від напівширини стрічки матриці та кількості використаних GPU

З графіків на цьому рисунку ми бачимо, що при збільшенні напівширини стрічки матриці від 3000 і більше отримано найбільше прискорення на багатовузловому гібридному комп'ютері (кількість GPU більше двох).

При використанні матриці з меншою напівшириною стрічки отримане прискорення набагато менше. Це пов'язано з недостатньою завантаженістю обчислювальних пристроїв та великою кількістю міжпроцесорних обмінів.

На рисунку 3.4 показано результати дослідження алгоритму при розв'язуванні АПВЗ для матриць Bone010 з Флоридської колекції, використовуючи блоки різного порядку.

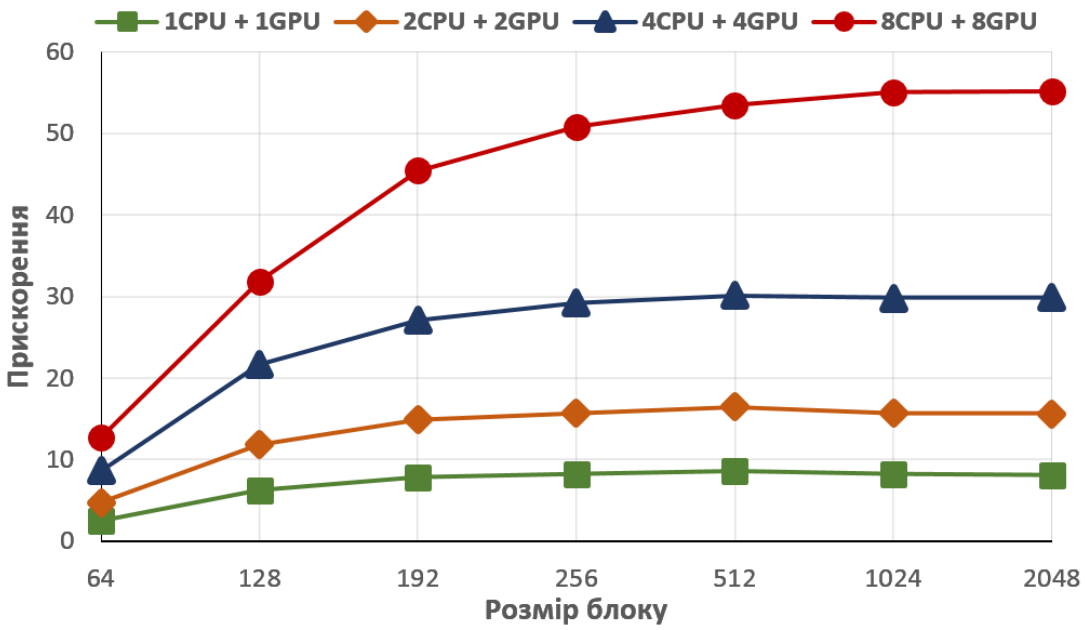

Рисунок 3.5 - Залежність прискорення гібридного алгоритму методу ітерацій на підпросторі від порядку блоків

З представленого на рисунку графіку видно, що при збільшенні розміру блоку до 1024 на гібридному комп'ютері було отримано найбільше прискорення. При використанні блоків меншого розміру отримане прискорення набагато менше. Це пов'язано зі збільшенням кількості міжпроцесорних обмінів, неефективним використанням кеш-пам'яті центрального процесора, а також збільшенням кількості викликів програмних функцій матрично-векторних обчислень.

Нижче можете спостерігати результат розв'язування задачі з наступними вихідними даними:

- порядок матриць – 12282;
- напівширина стрічки матриці A – 6212;
- напівширина матриці B – 71;
- обсяг пам'яті – 2 Gb.

Розв'язування задачі проведено на паралельних комп'ютерах різної архітектури:

- комп'ютер гібридної архітектури СКІТ-4 з CPU – серії Intel(R) Xeon(R) E5-2600; у вузлі – 2 CPU по 8 ядер; графічні процесори – Nvidia Tesla M2075;

- кластер СКІТ-5.5 AI з процесорами Intel(R) Xeon(R) E5-2695 v4, у вузлі 2 CPU по 18 ядер, графічні процесори – Nvidia Tesla K20.

Часові результати розв'язування різними алгоритмами:

- послідовним алгоритмом час розв'язування – 22 хв. 38 сек;
- паралельним алгоритмом на СКІТ-4, використовуючи 2 вузли (4 процесори по 8 ядер), час розв'язування – 2 хв. 13 сек;
- гібридним алгоритмом на СКІТ-4: при використанні одного CPU та GPU – 1 хв. 30 сек, при використанні двох CPU та GPU – 0 хв. 47 сек;
- паралельним алгоритмом на СКІТ-5.5 AI використовуючи 1 вузол, (2 процесори по 18 ядер) – 1 хв. 12 сек;
- гібридним алгоритмом на СКІТ-5.5 AI [9]: при використанні одного CPU та GPU – 0 хв. 35 сек.

Отже, за розробленим гібридним алгоритмом розв'язування АПВЗ **отримано такі прискорення** у порівнянні з послідовною версією алгоритму: на MIMD-комп'ютері – в 9 раз; на гібридному комп'ютері – в 15 і 28 разів, використовуючи один і два GPU відповідно; на гібридному комп'ютері СКІТ-5.5 AI – в 38 раз.

Нижче наведено фрагмент протоколу розв'язування задачі з такими вихідними даними: порядок матриць – 12282; напівширина стрічки матриці A – 6212 та матриці B – 71; обсяг пам'яті – 2 Gb.

Фрагмент протоколу розв'язування задачі

```
PROBLEM:
        Solving of Partial generalized eigenvalue problem
        for Band symmetric matrices
Hybrid algorithm of Subspace iterations

INPUT PARAMETERS:
        Order of matrices                = 12282
        Bandwise of matrix A             = 12423
        Bandwise of matrix B             =    141
        Number of minimal eigenvalues
                             to calculate  = 10

Exact eigenvalues

    2.467401226414946e+00    1.233703835857750e+01    2.220662009449054e+01
```

```
    3.207625722665309e+01    4.194634252990407e+01    6.168510614945586e+01
    6.168556139797965e+01    7.155474328161840e+01    9.129649208060502e+01
    1.011640474529450e+02    1.110357109486806e+02    1.209029560217768e+02
    1.998603146687910e+02    2.985573753576974e+02    1.307725931539393e+02
    2.097299518009535e+02    1.505141970036459e+02    1.603818973252659e+02

Process of research and solving of the problem

Method: Subspace Iterations

       Matrix blocksize           = 64
       Number of process          = 64
       Number of GPU's            =  0

Problem solving: total time (sek) = 26.0033e+00

Results: SOLUTION WAS CALCULATED
         by 20 iterations

All calculated eigenvalues are minimal

FIRST 3 EIGENVALUES
Eigenvalues (calculated)     Estimates of Errors

    1.248665556779830e-01          1.893e-06
    1.248734577580778e-01          8.894e-07
    1.248765574182433e-01          2.837e-06

All calculated eigenvalues stored in the file result.out

# End of problem solving
```

Обчислені мінімальні власні значення дають можливість визначити величини критичних параметрів стійкості шарувато композитного матеріалу при стисненні поверхневим навантаженням.

Проведена апробація запропонованого гібридного алгоритму для різних даних задачі, використовуючи різні комп'ютерні архітектури, свідчить про перспективність напрямку розвитку алгоритмічного та програмного забезпечення для математичного моделювання стійкості конструкцій.

3.3 Висновки до розділу

Побудовано і досліджено гібридний алгоритм методу ітерацій на підпросторі для розв’язування часткової узагальненої АПВЗ (кілька власних значень) стрічкових симетричних матриць, на гібридних комп’ютерах, що поєднують багатоядерні центральні процесори з розподіленою пам’яттю та масивно-паралельні графічні прискорювачі.

Запропоновано ефективний розподіл даних між обчислювальними пристроями гібридного комп’ютера. Проведено дослідження прискорення та ефективності створеного алгоритму.

4 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

4.1 Особливості розробки паралельних програм для розв'язування АПВЗ на гібридних комп'ютерах

Як вже зазначалося у попередніх розділах дисертації, комп'ютери гібридної архітектури передбачають наявність комп'ютера MIMD-архітектури, що складається з обчислювальних вузлів, кожен з яких може мати декілька багатоядерних процесорів та свою локальну пам'ять [1]. Кожен процесор має доступ тільки до локальної пам'яті свого вузла. Обчислювальні вузли об'єднуються між собою деяким комунікаційним середовищем, тому доступ до даних, які розміщені в пам'яті інших вузлів, виконується довше. Кожен вузол доповнюється одним або кількома графічними прискорювачами SIMT-архітектури [4], що відзначаються відносно високою продуктивністю та енергоефективністю.

Таким чином, у паралельній програмі, яка реалізує гібридний алгоритм, передбачається два рівня паралелізму: паралелізм між підзадачами на CPU (за допомогою MPI) та паралелізм математичних операцій на GPU (за допомогою технології CUDA) при реалізації кожної підзадачі.

При створенні гібридних алгоритмів насамперед необхідно визначити окремі підзадачі та проаналізувати можливість їх розпаралелити, пріоритетність, обчислювальні пристрої, схеми декомпозиції для ефективного виконання.

Схема декомпозиції даних між процесами на CPU гібридного комп'ютера є одним з основних факторів, які впливають на ефективність як алгоритму, так і програми. Використання розподіленої пам'яті MIMD-комп'ютера при розпаралеленні обчислень на CPU створює деякі проблеми щодо міжпроцесорних пересилок даних через комунікаційне середовище, які за тривалістю можуть значно перевершувати час виконання арифметичних операцій. Крім того, збалансованість завантаження процесорів також залежить від способів розподілу даних між процесами, зберігання (подання) і обробки масивів даних задачі (наприклад, матриць та векторів). З метою ефективного виконання паралельних обчислень потрібно прагнути до рівномірного

завантаження в кожен момент часу всіх процесорів, що беруть участь у розв'язуванні задачі, мінімізуючи час їхнього перебування в стані очікування.

При розробці гібридних програм для реалізації алгоритмів, що розглядаються, застосовувався двовимірний блочно-циклічний розподіл даних, при якому блоки містять не цілі рядки матриці, а їх частини. Такий підхід дає можливість у гібридних програмах, з урахуванням технічних властивостей комп'ютера та обсягу задачі, вибирати величину блоків таким чином, щоб обробка кожного окремого блоку здійснювалась в межах кеш-пам'яті CPU. Крім того, блочно-циклічний розподіл даних між процесами дає можливість рівномірно розподіляти інформацію між процесами та задіювати їх до закінчення обчислень [1].

Віртуальна топологія міжпроцесорних зв'язків та способи обміну даними між процесами також мають великий вплив на ефективність виконання гібридних алгоритмів, а отже, і програм. При виконанні задачі за гібридним алгоритмом виконуючі паралельні процеси можуть взаємодіяти попарно за допомогою обмінів типу "точка-точка" або групою з використанням колективних обмінів.

Найчастіше виконуються такі види розпаралелення:

– мультирозсилка, за якою один з процесів взаємодіючої групи розсилає дані всім процесам цієї групи;

– мультизбирання числа, при якому усі процеси взаємодіючої групи передають рівні порції даних одному з процесів цієї групи, виконуючи при цьому математичні операції, наприклад, додавання, вибір максимального або мінімального значення і тому подібне;

– мультизбирання вектора, під час якого всі процеси взаємодіючої групи передають порції даних одному з процесів групи, де з прийнятих даних формується новий вектор.

Всі ці дії виконуються в гібридній програмі за допомогою відповідних функцій системи розпаралелення MPI за такою схемою:

– визначається комунікатор (комунікаційне середовище), в якому будується ефективна топологія (логічна конфігурація зв'язків між MPI-процесами) для конкретного алгоритму та задаються логічні номери MPI-процесів;

– розподіляються між MPI-процесами вихідні дані та виділяється необхідний обсяг пам’яті для масивів;
– на процесах одночасно виконуються обчислення (при цьому на різних процесах можуть виконуватися різні обчислення) та здійснюються різні способи мультиопераций над отриманими результатами.

Виконання обчислень на графічних процесорах здійснюється за допомогою технології CUDA, що надає можливість використовувати 6 видів пам’яті, кожна з яких має своє призначення. При цьому зв’язок між CPU та GPU (пересилка інформації) здійснюється лише через глобальну (*global*) пам’ять, яка дуже повільна, а обчислення виконуються на розділеній (*shared*) швидкій пам’яті. Оскільки ця пам’ять невелика, то блоки, які виявилися найбільш придатними для кеш-пам’яті CPU, не помістяться в ній. Тому у програмі, яка виконується на процесі CPU, необхідно матрицю розбити на смуги по 16 стовпчиків, які копіюються в глобальну пам’ять відповідного процесора GPU. Розбивши смуги матриць на блоки (підматриці) 16×16, можна ефективно виконувати матрично-векторні операції цих підматриць на *shared*-пам’яті [4].

На процесорах GPU однотипні обчислення розпаралелюються на безліч потоків, забезпечуючи високу швидкодію. Якщо обсяг обчислень на GPU буде значно перевищувати обсяги даних, які пересилаються при цьому між CPU та GPU для подальших обчислень, то можна значно підвищити ефективність реалізації алгоритму. В іншому випадку обчислення на GPU не перекриють накладні витрати на пересилку. Також має бути визначений максимальний розмір масиву, що обчислюється на GPU. При малому розміру масиву не буде достатньої кількості потоків для покриття витрат на зчитування із пам’ятті. Отже, процесори на GPU мають бути якомога більше заповненими.

Дані від CPU можна копіювати на глобальну пам’ять GPU викликом функції *cudaMemcpy*. Необхідний обсяг глобальної пам’яті виділяється, у програмі, що працює на CPU, викликавши функцію *cudaMalloc*. З процесорів GPU цю функцію викликати не можна. Таким чином, типова гібридна програма, що реалізує гібридний алгоритм, складається з двох частин: головна частина програми (*host*-програма) − для

запуску MPI-процесів на CPU та для керування обчисленнями на GPU, а також CUDAпрограма (*kerner*), що реалізує багатопоточні однотипні обчислення з великими обсягами даних на *shared*-пам’яті GPU (схема реалізації обчислень – рисунок 4.1).

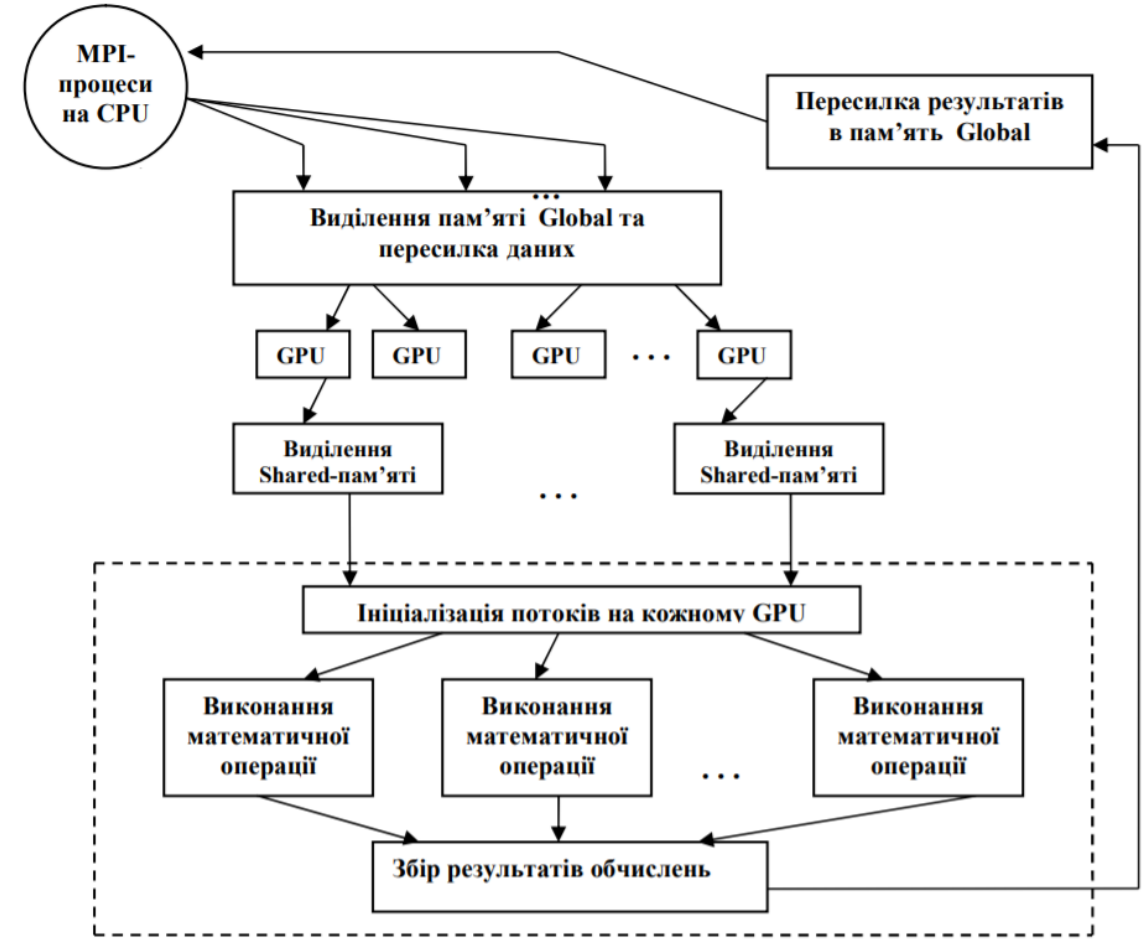

Рисунок 4.1 - Схема виконання обчислень в гібридній програмі

4.2 Засоби розробки

Розробка проекту проводиться за допомогою мови програмування Python.

Дана мова буде зручна як у реалізації алгоритмів так і при створенні частини нейромережі. Інтерфейс та взаємодія користувачів забезпечується через консольний термінал. Реалізація проекту використовуе такі технології, як TensorFlow, CUDA, Keras та інші.

Далі буде розглянуто переваги використаних технологій та мов програмування.

Python. Широко використовувана інтерпретована, динамічна, мова програмування високого рівня. Його філософія дизайну підкреслює читабельність

коду, а його синтаксис дозволяє програмістам, висловити поняття в меншій кількості рядків коду, ніж це було б можливо в інших мовах програмування.

Python підтримує кілька парадигм програмування, в тому числі об'єктно-орієнтований, імперативний, функціональний та процедурний підхід до програмування. Він має динамічну систему типів, автоматичне керування пам'яттю, те широку стандартну бібліотеку. Крім цього спільнотою відкритого вихідного коду, розробляються велика кількість допоміжних бібліотек для цієї мови програмування. Однією з таких бібліотек є TensorFlow за допомогою якої реалізована побудова нейромережі.

TensorFlow ™ - це бібліотека програмного забезпечення з відкритим кодом для численних обчислень з високою ефективністю. ЇЇ гнучка архітектура дозволяє легко розгортати обчислення на різних платформах (процесори, графічні процесори, TPU), а також від настільних комп'ютерів до кластерів серверів для мобільних пристроїв. Спочатку розроблена дослідниками та інженерами команди Google Brain в організації AI Google, вона має сильну підтримку для машинного навчання та глибокого навчання, а гнучкі числові обчислення використовуються в багатьох інших наукових областях.

CUDA - це модель паралельної обчислювальної платформи та інтерфейсу прикладного програмування (API), розроблена компанією Nvidia. Це дозволяє розробникам програмного забезпечення та розробникам програмного забезпечення використовувати графічний процесор (GPU) з підтримкою CUDA для обробки загального призначення - підхід, який називається GPGPU (загальнооб'єдноване обчислення на блоках графічної обробки). Платформа CUDA - це програмний рівень, який забезпечує прямий доступ до набору віртуальних наборів графічних процесорів та паралельних обчислювальних елементів для виконання обчислювальних ядер.

Платформа CUDA призначена для роботи з мовами програмування, такими як C, C ++ і Fortran. Ця доступність спрощує паралельне програмування фахівцями використовувати ресурси GPU, на відміну від попередніх API- інтерфейсів, таких як Direct3D та OpenGL, які вимагають високих навичок графічного програмування. Крім того, CUDA підтримує схеми програмування, такі як OpenACC та OpenCL. Коли він

вперше був представлений компанією Nvidia, назва CUDA була абревіатурою Compute Unified Device Architecture, але Nvidia згодом відмовилась від використання акроніму.

Keras - бібліотека нейронних мереж із відкритим кодом, написана на Python. Вона здатна працювати з TensorFlow, Microsoft Cognitive Toolkit, Theano або MXNet. Розроблена для швидкого експерименту з глибокими нейронними мережами, вона зосереджена на тому, щоб бути зручною, модульною та розширюваною. Вона була розроблена як частина дослідницької роботи проекту ONEiros (Open-ended Neuro-Electronic Intelligent Robot Operating System), і її основним автором і супроводжувачем є Франсуа Шолле, інженер Google.

У 2017 році команда Google TensorFlow вирішила підтримати Keras у базовій бібліотеці TensorFlow.Keras задумано як інтерфейс, а не самостійний механізм навчання. Він пропонує більш високий рівень інтуїтивно зрозумілого набору абстракцій, що полегшує розробку глибоких моделей навчання незалежно від використовуваного обчислювального бекенда.

4.3 Вимоги до технічного забезпечення

Представлений програмний продукт являє собою программу, для роботи якої, необхідна робоча странція з наступним технічним забезпеченням:

а) технічне забезпечення:

1) процесор з тактовою частотою не нижче 2 ГГц;
2) об'єм оперативної пам’яті не менше 2 Гб;
3) ПЗУ типу SSD або HDD об’ємом не менше 20 ГБ;

б) програмне забезпечення:

1) операційна система – Windows версії не нижче Windows 7;
2) інтерпретатор Python не нижче версії 3;
3) Наступні бібліотеки для Python : TensorFlow, scipy, pyplotlib, Keras, pillow.

Для для тренування нейромережі потрібне наступне програмне та технічне забезпечення:

а) комп'ютер з характеристиками не нижче:

1) процесор з тактовою частотою 2.5 ГГц;
2) оперативна пам’ять 4 Гб;
3) ПЗУ типу SSD або HDD об’ємом не менше 20 ГБ;
4) відео прискорювач Nvidia з підтримкою технології CUDA (GeForce GTX 590 та краще).

б) програмне забезпечення:

1) операційна система – Windows версії не нижче Windows 7;
2) інтерпретатор Python не нижче версії 3;
3) наступні бібліотеки для Python : TensorFlow, scipy, pyplotlib, Keras, pillow;
4) програмний пакет CUDA.

в) комп’ютерна периферія, до складу якої входить:

1) монітор;
2) мишка;
3) клавіатура.

4.3 Висновки до розділу

Визначено програмне та технічне забезпечення, яке було використано при написанні даної роботи, розглянуті особливості розробки паралельних програм для розв’язування АПВЗ на гібридних комп’ютерах. Виділено вимоги до технічного забезпечення.

5 НЕЙРОННА МЕРЕЖА

На теперішній час розроблено велику кількість паралельних і гібридних алгоритмів для розв'язування задач лінійної алгебри з розрідженими матрицями. В роботах [2, 3] розроблено та досліджено «хмарочосну» схему зберігання розрідженої матриці та паралельний алгоритм розв'язування СЛАР з розрідженими матрицями на основі $U^{T}DU$-розвинення та досліджено ефективність двовимірного блочно-циклічного алгоритму методу Гаусса, у [4] паралельні алгоритми розв'язування СЛАР з розрідженими матрицями апробовано на задачах аналізу міцності будівельних конструкцій. В [5] розроблено та досліджено плитковий алгоритм факторизації для гібридних систем з декількома GPU.

У [6] отримано оцінки прискорення і ефективності гібридного алгоритму розв'язування лінійних систем з розрідженими матрицями на основі блочного LL^{T} розвинення, проведено апробацію алгоритму на низці матриць, які виникають при моделюванні процесів різної природи. У [7] досліджено гібридні алгоритми для розв'язування часткової узагальненої проблеми власних значень розріджених матриць та проведено експерименти, які свідчать про ефективність нових алгоритмів. Такий аналіз показує очевидну ефективність комп'ютерного алгоритму від структури матриць задач.

5.1 Постановка проблеми

Однією з проблем, яка виникає при розв'язанні практичних задач для довільних розріджених матриць апріорі не визначеної структури полягає у виборі ефективного алгоритму розв'язання задачі. Оскільки нам наперед невідомо структура матриці, її тип і характеристики є ризик розв'язувати задачу алгоритмом, який є мало ефективним для цієї задачі. В статті пропонується підхід, який за допомогою нейронної мережі дозволить автоматизувати процес класифікації типу матриці, і в результаті вибрати найбільш ефективний алгоритм розв'язання задачі. Це дозволить скоротити час розв'язання прикладних задач і більш ефективно використовувати ресурси обчислювальних систем, зокрема і високопродуктивних комп'ютерів СКІТ [8].

Для вирішення проблеми пропонується підхід який можна описати наступним чином:

- отримання матриці системи;
- візуалізація матриці;
- визначення типу матриці за допомогою нейронної мережі;
- за результатом роботи нейронної мережі визначаємо алгоритм, ефективний для типу матриці;
- розв’язання задачі відповідним методом.

Розглянемо задачу класифікації об’єктів. В загальному випадку її можна описати наступним чином. Нехай X - множина описів об'єктів, Y - множина номерів (чи назв) класів. Існує невідома цільова залежність – відображення $y^*: X \to Y$, значення якої відомі лише на елементах скінченної навчальної вибірки $X^m = \{(x_1, y_1), \ldots\ldots (x_m, y_m)\}$. Потрібно побудувати алгоритм $\alpha: X \to Y$, здатний класифікувати довільний об'єкт $x \in X$.

Вхідні дані для задачі класифікації можуть бути наступними:

- характеристичний опис. Кожен об'єкт описується набором своїх характеристик, які називаються ознаками. Ознаки можуть бути числовими або нечисловими;
- матриця відстаней між об'єктами. Кожен об'єкт описується відстанями до всіх інших об'єктів навчальної вибірки. З цим типом вхідних даних працюють деякі методи, зокрема, метод найближчих сусідів, метод потенційних функцій;
- часовий ряд або сигнал є послідовність вимірів у часі. Кожен вимір може представлятися числом, вектором, а в загальному випадку — характеристичним описом досліджуваного об'єкта в цей час часу;
- зображення або відеоряд;

- зустрічаються і складніші випадки, коли вхідні дані представляються у вигляді графів, текстів, результатів запитів до бази даних, і т. д. Як правило, вони приводяться до першого або другого випадку шляхом попередньої обробки даних та вилучення характеристик.

За типами класів розглядають:

- двокласова класифікація. Найпростіший в технічному відношенні випадок, який служить основою для вирішення складніших завдань;
- багатокласова класифікація. Коли число класів досягає багатьох тисяч (наприклад, при розпізнаванні ієрогліфів або злитого мовлення), завдання класифікації стає істотно важчим;
- непересічні класи;
- пересічні класи. Об'єкт може належати одночасно до декількох класів;
- нечіткі класи. Потрібно визначати ступінь належності об'єкта кожному з класів, звичайно це дійсне число від 0 до 1.

В машинному навчанні завдання класифікації вирішується, як правило, за допомогою методів штучної нейронної мережі при постановці експерименту у вигляді навчання з учителем. Одним з ефективних типів нейронних мереж є згорткові нейронні мережі [9].

В нашому випадку ми будемо працювати з зображеннями і нечіткими класами.

5.2 Структура нейронної мережі

Для розв’язання задачі визначення типу матриці побудовано згорточку нейронну мережу “Sparse Matrix Vision”. Нижче розглянемо структуру розробленої мережі.

Основні базові елементи мережі наступні:

- вхідний шар;
- згортковий шар;

- шар підвибірки;
- повнозв’язний шар;
- вихідний шар.

На рисунку 4.1. показано логічну схему згорткової мережі

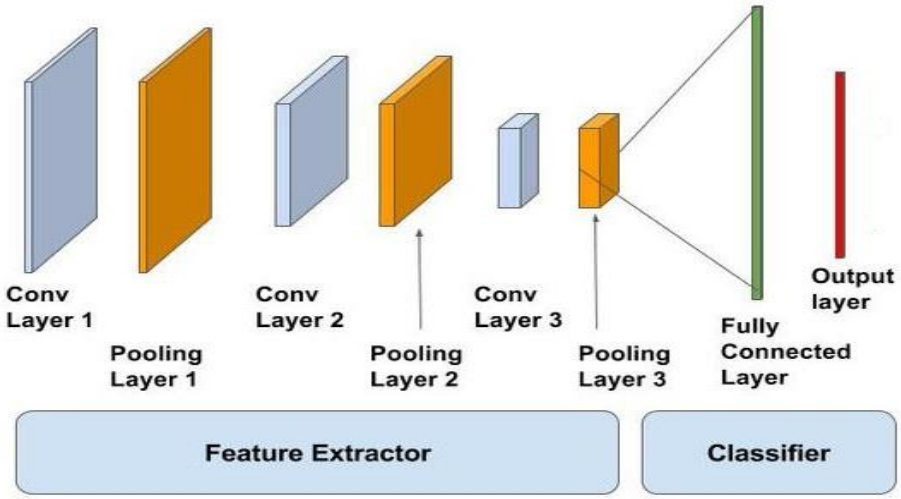

Рисунок 5.1 - Структура нейронної мережі “Sparse Matrix Vision”

Вхідний шар. Вхідними даними для мережі є зображення типу JPEG, розміром 224x224 пікселів. Зображення розбивається на 3 канали: червоний, зелений, синій. Вхідний шар враховує двовимірну топологію зображення і складається з кількох карт, де кожна карта відповідає кожному каналу зображення. Вихідні дані кожного пікселя нормалізується в діапазоні від 0 до 1, за формулою

$$f\left(p, min, max() \frac{p - min}{max - min,}\right) \tag{5.1}$$

де f - функція нормалізації, p - значення кольору конкретного пікселю, min - мінімальне значення пікселя, max - мінімальне значення пікселя.

Згортковий шар. Згортковий шар представляє з себе набір карт (інша назва – карти ознак, в побуті це звичайні матриці), у кожної карти є синаптичне ядро (в різних джерелах його називають по-різному: сканує ядро або фільтр). Розмір у всіх карт згорткового шару – однакові і обчислюються за формулою

$$(w,h) = (mW - kW + 1, mH - kH + 1), \quad (5.2)$$

де (w,h) - розмір, що обчислюється, mW - ширина попередньої карти, kW - ширина ядра, mH - висота попередньої карти, kH - висота ядра.

Ядро являє собою фільтр або вікно, яке ковзає по всій області попередньої карти і знаходить певні ознаки об'єктів. Також розмір ядра вибирається таким, щоб розмір карт згорткового шару був парним, це дозволяє не втрачати інформацію при зменшенні розмірності шару підвибірки.

Значення ваг ядер задаються випадковим чином в області від -0.5 до 0.5. Ядро пробігає по попередній карті і виконує операцію згортки за формулою

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l] * g[k,l], \quad (5.3)$$

де f - вихідна матриця зображення, g - ядро згортки.

При цьому за рахунок крайових ефектів розмір вихідних матриць зменшується за формулою:

$$x_j^l = f\left(\sum_i x_i^{l-1} * k_j^l + b_j^l\right), \quad (5.4)$$

де $f(\quad)$ - функція активації, x_j^l - вихід шару l, b^l - коефіцієнт зсуву шару l для карти ознак j, k_j^l - ядро згортки j карти ознак шару l, * - операція згортки входу шару l, з ядром згортки k.

Шар підвибірки. Шар також, як і згортковий має карти, але їх кількість збігається з попереднім (згортковим) шаром. Ціль шару – зменшення розмірності карт попереднього шару. Якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки докладне зображення вже не потрібно, і воно ущільнюється до менш докладного.

Зазвичай в подвиборочном шарі застосовується функція активації *RelU*. Операція підвибірки (або MaxPooling-вибір максимального) відповідно до

рисунок 5.2.

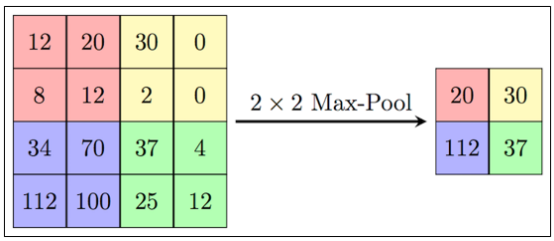

Рисунок 5.2 - Операція “Max-Pool”

Шар може бути описаний формулою

$$x^l = f(a^l * subsample(x^{l-1}) + b^l), \qquad (5.5)$$

де $f(\ \ )$ - функція активації, x^l - вихід шару l, a^l, b^l - коефіцієнт зсуву шару l $subsample(\ \ )$ - функція вибору локальних максимальних значень.

Повноз'єднаний шар (fully-conected). Після шарів згортки ми отримаємо безліч карт ознак. Їх з'єднаємо в один вектор і цей вектор подамо на вхід fully connected мережі.

Формула для fc-шару (fully connected) виглядає так:

$$x_i^l = \sum_{k=0}^{m} w_{ki}^l y_k^{l-1} + b_i^l \qquad \forall i \in (0, \dots, n) \qquad (5.6)$$

Вихідний шар. Вихідний шар пов'язаний з усіма нейронами попереднього шару. Кількість нейронів відповідає кількості розпізнаваних класів, в нашому випадку 5:

- профільна матриця;
- стрічкова матриця;
- блочно-діагональна матриця;
- блочно-діагональна матриця з обрамленням;
- матриця хмарочосного виду.

У нейронній мережі використовуються наступні функції активації:

$RelU\ f(s) = max(0, s)$;

$Sigmoid\ f(s) = \frac{1}{1+e^{-s}}$.

Представлення матриці в різних масках кольорів. На рисунку 5.3 та рисунок4.4 можна спортерігати як нейронна мережа «бачить» нашу матрицю після застосування різних фільтрів. Чим інтенсивніший колір - тим сильніша ознака.

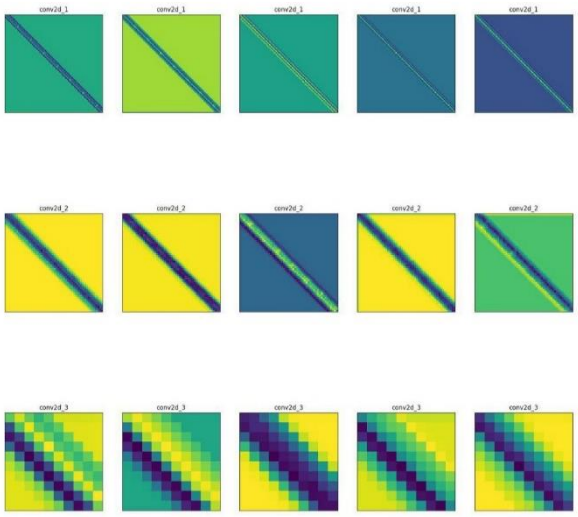

Рисунок 5.3 - Представлення матриці в різних масках кольорів

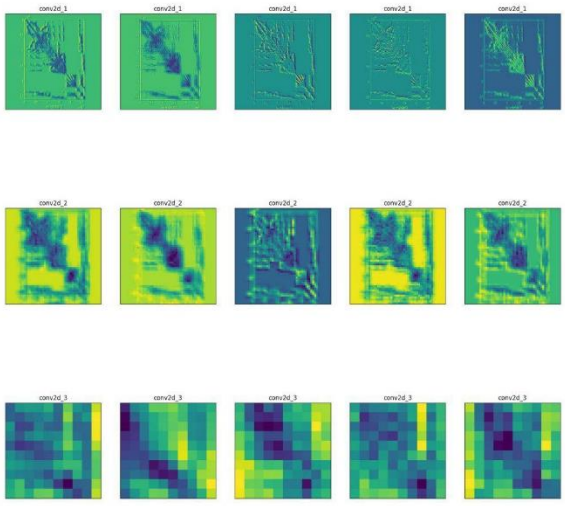

Рисунок 5.4 - Представлення матриці в різних масках кольорів

Програмна реалізація і чисельний експеримент. Для проведення чисельного експерименту і навчання нейронної мережі створено віртуальне оточення Python [10] з встановленими програмними пакетами Keras [11], Tenzorflow [12].

Для розрахунків використано вузол СКІТ з наступними характеристиками:

- 2 центральні процесори Intel Xeon E5-2600 з частотою 2.6 ГГц;
- інтегрований із загальним сховищем даних кластерного комплекса обсягом 200 ТБ;
- мережа передачі даних між вузлами Infiniband FDR 56 Гбіт/с.
- 128 ГБ оперативної пам’яті.

Для навчання нейронної мережі використано набір з 2500 зображень. Деякі зображені на рисунку 1.

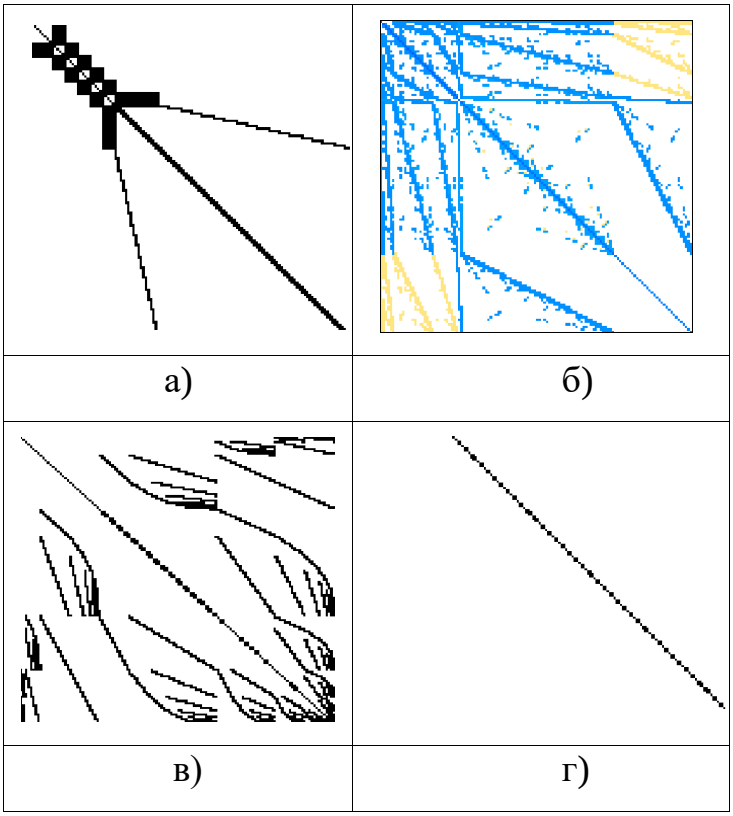

Рисунок 5.5 - Профілі наповненості ненульовими елементами розріджених матриць (а - G3_circuit, б - Dubcova2, в - parabolic_fem, г - apache2)

В результаті навчання нейронної мережі отримано наступні графіки точності нейроної мережі та графік втрат даних (рисунок 5.6 - 5.7).

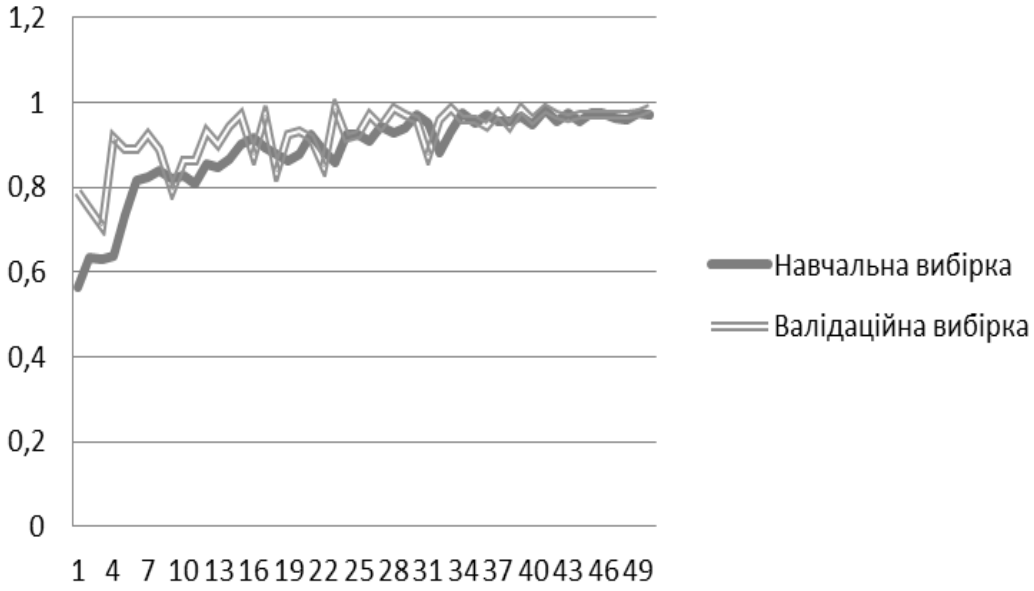

Рисунок 5.6 - Точність нейронної мережі

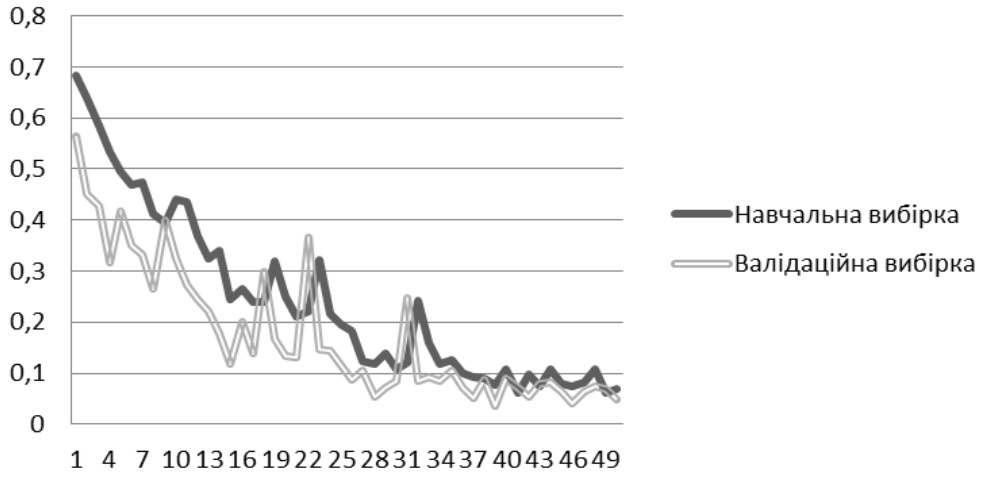

Рисунок 5.7 - Втрати нейронної мережі

Приклад виводу нейроної мережі:

0.8869001 0.00093775 0.00008755 0.00000035 0.0000435.

Елементи вектора показуюють ймовірність приналежності зображення до відповідного класу. Зокрема, для випадку приведеного вище можна сказати, що з ймовірністю 88% матриця профільна.

5.3 Висновки до розділу

Запропоновано підхід для автоматизації вибору алгоритму розв’язання прикладних задач. Розроблено і навчено нейронну мережу для розпізнавання типу розрідженої матриці. Проведені чисельні експерименти на вузлі суперкомп’ютера СКІТ показали високу точність класифікації типу матриці.

6 РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

6.1 Опис ідеї проекту

З метою визначити, чи зможе розроблювальний продукт вийти на ринок та конкурувати із продуктами, що вже посіли на ньому своє місце, у цьому розділі буде проведений аналіз стартап проекту.

Призначенням проекту є спрощення проектування конструкцій.

Сутністю розробки є створення інструменту для спрощення та пришвидшення знаходження слабких місць в конструкції.

Цільовою аудиторією для використання цього продукту є будівельні компанії та конструкторські бюро, які займаються створенням інженерних проектів та розраховують слабкі місця в конструкціях.

Основною вигодою використання цього продукту допоможе швидко знайти слабкі місця в створених проектах конструкцій.

Для того, щоб мати цілісне уявлення про зміст ідеї, а також можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів, опишемо зміст ідеї стартап проекту, напрямки можливого застосування, а також вигоди для користувача у таблиці 6.1.

Таблиця 6.1 — Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Аналіз стійкості будівельної конструкції шляхом визначення візуалізованої матриці жорсткості за допомогою нейронної мережі, а також розрахунок стійкості будівельної конструкції найефективнішим алгоритмом	Будівництво	Автоматизований вибір алгоритму розв'язання задачі та оптимізоване використання комп'ютерних ресурсів . До того ж, алгоритм розв’язання задачі використовує графічні процесори комп’.тера, що зменшує час розрахунків.

Отже, ідея створення системи визначення слабких місць в конструкціїях є досить актуальної з огляду на те що кожного дня будуються безліч конструкцій різного призначення, від житлових будинків до великих індустріальних обєктів.

З огляду на те що для кожної конструкції розраховується склад матеріалів, потрібно точно визначити стійкість конструкції при будь якому впливу на цю конструцію.

Наступним кроком буде проведення аналізу потенційних техніко-економічних переваг ідеї, у порівнянні із тим, що пропонують конкуренти: визначити перелік техніко-економічних властивостей та характеристик ідеї; визначити попереднє коло конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та провести збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до переліку, визначеного вище; провести порівняльний аналіз показників: для власної ідеї визначити показники, що мають:

- гірші значення (W, слабкі);
- аналогічні (N, нейтральні) значення;
- кращі значення (S, сильні) (табл. 6.2).

Після детального пошуку у мережі інтернет інструментів подібних до запропонованої системи був знайдений два конкуренти: RFEM та RSTAB.

У таблиці 6.2 наведемо порівняльний аналіз сильних, слабких та нейтральних характеристик ідеї проекти у порівнянні із зазначеними конкурентами.

Таблиця 6.2 — Визначення сильних, слабких та нейтральних характеристик ідеї проекту

Ідея	**(потенційні) товари/концепції конкурентів**			**W (слабка сторона)**	**N (нейтральна сторона)**	**S (сильна сторона)**
	Мій проект	RFEM	RSTAB			
Вартість обслуговування	Відносно невелика	Не знайдено інформації	Не знайдено інформації		+	

Продовження таблиці 6.2

Вартість експлуатації	Відносно невелика	Не знайдено інформації	Не знайдено інформації		+	
Безвідмовність	За рахунок того, що система має багато рівнів а також, містить в собі композицію серверів ймовірність виходу всієї системи разом з ладу є дуже низькою	Досить висока	Досить висока			+
Ремонтопридатність	За рахунок того, що система розгортається на на комп'.ютері, відсутня будь-яка проблема з	Досить висока	Досить висока			+

Продовження таблиці 6.2

	ремонтом, адже ремонтувати прийдеться лише комп’ютер					
Ремонтопридатність	Беручи до уваги те, що дана система є стартапом, то на момент її запуску, вона буде розвиватись в більшій мірі за рахунок ентузіазму команди, і також можливих інвесторів, у вигляді майбутніх користувачів системи , а саме рекламодавців	Досить висока	Досить висока			+

Продовження таблиці 6.2

Зручність користування	Розроблений дизайн на основі загальноприйнятих паттернів, та найкращих практик UX	Висока	Висока			-
Простота освоєння	За рахунок того, що система розроблена на основі загальноприйнятих та звичних практик, простота освоєння є дуже високою	низька	низька			-
Дизайн ресурсу	Не достатньо опрацьований, за рахунок не дуже високої відповідності	Висока	Висока	+		

Продовження таблиці 6.2

	сучасним тенденціям					
Відповідність патернами дизайну	низька	Висока	Висока		+	
Відповідність тенденціям дизайну	низька	Висока	Висока		+	
Безпека даних користувача	Висока, так, як це одна з основних вимог до системи, що місять в собі дані про користувача	Дуже висока	Дуже висока			+

Продовження таблиці 6.2

Відмовостійкість системи	Враховуючи те, що система має багато рівнів, та являє собою композицію серверів, і також має періодичні задачі з дампуванням бази даних, то існує дуже низька ймовірність виходу всієї системи з ладу одчасно, з втратою якихось даних користувача тощо	Дуже висока	Дуже висока			+
Підтримка оновлень	Відсутня	Присутня	Присутня	+		

На основі визначених сильних, слабких та нейтральних сторін ідеї, можна провести аналіз та зрозуміти її конкурентоспроможність.

Отже, згідно наведеної таблиці можна зазначити, що розроблювальний проект має деякі переваги над системами-аналогами (своїми конкурентами), а саме: аналіз популярних електронних ресурсів на предмет використання патернів психологічного впливу та відсутність обмеження за обсягом введених даних.

Втім, на даному етапі розробки не планується включати у систему велику кількість інтеграцій з різними системами через брак часу. Але при необхідності інтеграції можуть бути реалізовані. Також, через певні обмеження використаних підходів неможливо виділяти конкретні ділянки тексту що відовідають патернам психологічного впливу. Але даний функціонал також не реалізований у конкурента.

6.2 Технічний аудит ідеї проекту

В рамках даного розділу проведемо аудит технологій, за допомогою яких можна реалізувати ідею проекту[11]. Співвідношення ідеї проекти із технологією реалізації наведено у таблиці 6.3.

Таблиця 6.3 — Технологічна здійсненність ідеї проекту

Ідея проекту	**Технології її реалізації**	**Наявність технологій**	**Доступність технологій**
Збір даних для матриці	Відкриті дані про стан будівельних конструкцій	Наявна	Доступна

Продовження таблиці 6.3

Тренування нейроної мережі	Навчання нейронної мережі класифікації зображень на відомих візуалізованих матрицях різних типів	Потребує розробки	Використовує ті технології, та засоби, які є вільними для використання
Класифікація зображення матриці	Використання нейронної мережі для визначення типу матриці за її візуалізацією	Потребує розробки	Використовує ті технології, та засоби, які є вільними для використання
Розв'язання СЛАР	Використання розробленого гібридного алгоритму для знаходження власних значень системи	Потребує розробки	Використовує ті технології, та засоби, які є вільними для використання
Інформаційна система	Використання засобів розробки Front-end та Back-end частин	Потребує розробки	

Як можна судити із наведеної таблиці, розроблювальний проект потребує розробки нових технологій, адже використовує ті, вже наявні що є безкоштовними.

6.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначимо ті ринкові можливості, які можна використати під час виходу проекту на ринок, а також такі загрози, що можуть зашкодити підприємству.

Це дозволить спланувати можливі напрямки розвитку проекту, з урахуванням того стану, у якому знаходиться ринкове середовище. Результати наведемо у таблиці 6.4.

Таблиця 6.4 — Попередня характеристика потенційного ринку стартап-проекту

Номер пункту	Показники	Характеристика
1	Кількість головних гравців	2
2	Загальний обсяг продаж, грн/ум.од	невідома
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Присутні обмеження: 1. Необхідний рівень безпеки даних користувачів (GDPR)

Продовження таблиці 6.4

		2. Необхідний рівень відказостійкості системи, для збереження позицій 3. Необхідний рівень відповідності патернам UX, зручності використання та візуальної складової для збереження підтримки користувачів, а також зручності використання системи для людей з обмеженими можливостями
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	Невідома

Оцінка потенційного ринку розродлювального стартап-проекту показує, що умови для входу до ринку є не досить сприятливими, особливо враховуючи що на ринку наявні фірми, які давно на ньому перебувають та чиї системи використовуються дуже успішно.

6.4 Розробка маркетингової програми стартапу

У цьому розділі сформуємо маркетингову концепцію товару, що отримає кінцевий споживач.

Кінцевий споживач може отримати доступ дектопної версії застосунку. Для нових користувачів надається тріал-версія продукту сроком до 1 місяця, для подальшого використання необхідно внести щомісячну сплату. Клієнт може підв’язати свою кредитну картку, і плата буде зніматися автоматично.

Для початку визначимо основні переваги продукту, що розробляється у таблиці 6.5.

Таблиця 6.5 — Основні переваги використання розроблювального продукту

Потреба	Вигода від використання продукту	Ключові переваги перед конкурентами
Вибір отптимального алгоритму для розв'язання задачі	Зменшення часу розв'язування задачі, виключення випадків відмови системи через недоцільне використання ресурсів комп'ютера	Точний вибір алгоритму вирішення задачі
Раціональне використання ресурсів комп'ютера	Програма не використовує зайвих ресурсів комп'ютера	Менший час вирішення задачі

Таким чином можна бачити, що продукт має явні переваги над конкурентами, та відповідає потребам користувачів. Наступним кроком розробимо оберемо систему збуту. Інформацію надамо у таблиці 6.6.

Таблиця 6.6 — Система збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Підпис	Основні функції постачальника: Розробка - Тестування - Вдосконалення - Збут та підтримка	Перший рівень	Пряма, із пошуком клієнтури

Також розробимо стратегію маркетингових комунікацій, та візьмемо за її основу специфіку поведінки клієнта, та стратегії позиціонування. Стратегія наведена у таблиці 6.7.

Таблиця 6.7 — Стратегія маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікації	Ключові позиції для позиціювання	Концепція рекламного звернення

Продовження таблиці 6.7

Покращення якості наданих послуг, зменшення їх вартості	Інтернет ресурси, контекстна реклама, прес-релізи	Канал першого рівня	Покращення якості продукту, зручний інтерфейс

6.5 Розробка ринкової стратегії стартапу

Для опису ринкової стратегії, необхідно визначити групи потенційних користувачів (таблиця 6.8).

Таблиця 6.8 — Групи потенційних користувачів продукту

Назва групи	Готовність користування проектом	Приблизний попит в межах групи	Інтенсивність конкуренції	Складність входу до сегменту
Компанії-забудовники	Готові	Високий	Середня	Середня
Конструкторські бюро	Готові	Середній	Середня	Середня

Згідно з таблицею, основними користувачами даного продукту можуть стати журналісти та працівники медіа сфери, оскільки вони мають справу з інтернет-виданнями та аналізують багато інформації, отриманої з джерел Інтернету. Також можливими користувачами можуть стати звичайні користувачі Інтернету, які хочуть перевіряти інформацію на наявність пропаганди. Саме цим сегментам варто пропонувати системи автоматичної класифікації текстових даних на наявність пропаганди, а працюючи із кожним сегментом, варто розробляти план впливу на нього окремо.

Сформуємо базову стратегію розвитку у цільових сегментах та наведемо її у таблиці 6.9.

Таблиця 6.9 — Визначення базової стратегії розвитку

Альтернатива розвитку продукту	Стратегія охоплення ринку	Конкурентноспроможні позиції відповідно до альтернативи	Базова стратегія розвитку
Стратегія спеціалізації	Пропонування продукту потенційним клієнтам, можлива модифікація ПО під потреби конкретного споживача	Довготривалі стосунки із клієнтами	Стратегія диференціації

У якості базової стратегії розвитку оберемо стратегію диференціації — стратегію, у якій орієнтування йде на потреби користувача. У разі провалу такої ідеї, стратегію можна змінити на стратегію спеціалізації — тобто орієнтування на конкретну цільову групу.

Наступним кроком визначимо стратегію конкурентної поведінки на ринку, та наведемо її у таблиці 6.10.

Таблиця 6.10 — Стратегія конкурентної поведінки на ринку

Чи є проект “першопрохідцем” на цільовому ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Частково	Шукати нових споживачів	Так як продукт є “першопрохідцем” на ринку, всі характеристики товару будуть створюватися з нуля під потреби користувача.	Стратегія заняття конкурентної ніші

Отже, обраною стратегією є зайняття конкурентної ніші з охопленням декількох сегментів споживачів, а оскільки продукт не є першопрохідцем на ринку, то споживачів потрібно у більшості випадків забирати у існуючих конкурентів.

Визначена стратегія позиціонування продукту на ринку наведена у таблиці 6.11.

Таблиця 6.11 — Стратегія позиціонування продукту

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
Зменшення часу вирішення задачі алгоритмом; Зменшення використання ресурсів комп'ютера	Диференціації	Висока якість, налаштування на довготривалі стосунки із клієнтами	Швидкодія Клієнторієнтованість Точність класифікації

6.6 Висновки до розділу

Метою цього розділу є формування ідеї для розробки стартап проекту. Спочатку було проаналізовано існуючий аналог системи проектування конструкцій та проведений порівняльний аналіз з розроблювальним продуктом. Єдиний аналог має схожий функціонал, але було визначено переваги розроблювального алгоритма перед вже існуючим.

Була проведена експертиза можливості втілення проекту з технічної точки зору, під час якої не було виявлено жодного ризику, оскільки для розробки треба використати ті інструменти, які вже існують та є безкоштовними.

Також був побудований маркетинговий план та визначені стратегії розвитку продукту на ринку.

ВИСНОВКИ

У даній магістерській дисертації було створено програму для вибору алгоритму розв’язання задачі шляхом класифікації типу вхідної матриці за допомогою нейронної мережі, та створенно гібридний алгоритм для вирішення часткової алгебраїчної проблеми власних значень стрічкових симетричних додатно-означених матриць великої розмірності на багатоядерних комп’ютерах з графічними процесорами. Програмна реалізація системи написана мовою програмування Python на платформі PyQt на базі операційної системи Windows. Отримано оцінки ефективності та прискорення розробленого алгоритму. Проведено апробацію алгоритмічно-програмного забезпечення при розв’язуванні задачі стійкості композитного матеріалу, що зводиться до розв’язування часткової узагальненої проблеми власних значень. Першим етапом розробки цього проекту був аналіз предметної області. Були досліджені останні теоретичні роботи у даній сфері, виокремлено та використано в роботі результати найбільш перспективних із них. На основі проведених досліджень стало можливим зробити обґрунтований вибір типу нейронної мережі, її структури, найбільш ефективного методу використання графічних процесів для ресурсозатратних обчислень. Таким чином в результаті проведеної роботи була досягнута поставлена мета дослідження.

За матеріалами дисертації було опубліковано 3 наукові роботи: 1 стаття [29] та 2 тез доповідей на конференціях [30; 31]

ПЕРЕЛІК ПОСИЛАНЬ

1. О.М. Хіміч, О.В. Чистяков. Паралельні однокрокові ітераційні методи розв’язання алгебраїчної проблеми власних значень для розріджених матриць. Компьютерная математика, 2014, № 2, С. 81−88.

2. О.В. Чистяков. Про особливості розробки програмного забезпечення для розв’язання задач на власні значення з розрідженими матрицями на гібридних комп’ютерах. Компьютерная математика, 2015, № 1, С. 75−85.

3. О.М. Хіміч, В.М. Бруснікін, О.В. Чистяков. Гібридний алгоритм узагальненого методу спряжених градієнтів для проблеми власних значень з симетричними розрідженими матрицями. Математичні машини та системи, 2015, № 3, С. 3−13.

4. О.М. Хіміч, О.В. Попов, А.Ю. Баранов, О.В. Чистяков. Гібридний алгоритм розв’язування задач на власні значення для стрічкових матриць. Теорія оптимальних рішень, 2016, 86−95.

5. О.В Чистяков. Гібридний алгоритм методу ітерацій на підпросторі для розв’язання задачі стійкості конструкцій. Математичне та комп’ютерне моделювання. Серія фізико-математичні науки, Інститут кібернетики імені В.М. Глушкова НАН України, Кам’янець-Подільський національ. університет імені І. Огієнка. 2017, вип. 15, С. 255–260.

6. A.N. Khimich, A.V. Popov, O.V. Chistyakov. Hybrid algorithms for solving the algebraic eigenvalue problem with sparse matrix. Cybernetics and Systems Analysis. 2017, vol. 53, no 6, pp. 132–146. 7. О.М. Хіміч, О.В. Попов, Т.В. Чистякова, О.В. Рудич, О.В. Чистяков. Інтелектуальна система для дослідження та розв’язування задач на власні значення на паралельних комп’ютерах з процесорами Intel Xeon Phi. Штучний інтелект, 2017. № 2, С. 119–127.

7. А.Н. Гузь. Основы трехмерной теории устойчивости деформируемых тел. К.: Наук. Вища школа, 1986, 512 с.

8. A.N. Guz. Fundamentals of the Three-Dimentional Theory of Stability of Deformable Bodies. Berlin –Heidelberg – New York: Springer, 1999, 555 p.

9. А.Н. Гузь, В.А. Декрет. Модель коротких волокон в теории устойчивости композитов. Saarbrücken: LAP, 2015, 315 с.

10. А.Н. Гузь, Ю.В. Коханенко. Численное решение задач трехмерной теории устойчивости упругих тел. Прикладная механіка, 2004, том 40, № 11, С. 117–126.

11. A.N. Guz, V.A. Dekret, Yu V. Kokhanenko. Solution of plane problems of the three-dimension problems stability of a ribbon-reinforced composite. Int. Appl. Mech, 2000, Vol. 36, № 10, pp. 1317–1328.

12. А.Н. Гузь, В.А. Декрет. Модель коротких волокон в теории устойчивости композитов. Saarbrücken: LAP, 2015, 315 с.

13. A.N. Guz, V.A. Dekret, Yu V. Kokhanenko. Solution of plane problems of the three-dimension problems stability of a ribbon-reinforced composite. Int. Appl. Mech, 2000, Vol. 36, № 10, pp. 1317–1328.

14. Розен Б.У. Механика упрочнения композитов. В: «Волокнистые композитные материалы. – М.: Мир, 1967». – С. 54 – 94.

15. Розен Б.У., Дау Н.Ф. Механика разрушения волокнистых композитов. В: «Разрушение, т. 7, ч. 1. Разрушение неметаллов и композитных материалов. Неорганические материалы (стекла, горные породы, композиты, керамики, лед). – М.: Мир, 1976». – С. 300 – 366. 28.

16. Чамис К. Микроскопические теории прочности. В: «Композитные материалы, т. 5. Разрушение и усталость. – М.: Мир, 1978». – С. 106 – 166.

17. Черепанов Г.П. Механика разрушения композиционных материалов. – М.: Наука, 1983. – 298 с.

18. Babich I.Yu. and Guz A.N. Deformation instability of laminated materials // Sov. Appl. Mech. – 1969. – 5, N 5. – P. 488 – 491.

19. Budiansky H. Micromechanics // Composites and Structures. – 1983. – 16, N 1. – P. 3 – 13.

20. http://ufdjrw.blogspot.com/2019/03/blog-post_2743.html

21. TOP 500 Supercomputer Sites. [Електронний ресурс], Lists: 2018 (06), режим доступу: https://www.top500/org/

22. Воеводин В.В.,Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-

Петербург, 2002, 608 с.

23. Боресков А.В.,Харламов А.А. Основы работы с технологией CUDA.

М.: ДМК Пресс, 2010, 232 с.

24. Python. [Електронний ресурс],режим доступу: https://www.python.org/

25. Keras. [Електронний ресурс], режим доступу: https://keras.io/

26. Tensorflow. [Електронний ресурс],режим доступу: https://www.tensorflow.org/

27.В.А.Сидорук,П.С.Єршов,Д.О.Богурський,О.Р.Марочканич,Інтелектуалізація обчислень для задач математичного моделювання складних процесів і об’єктів, Компьютерная математика.

28. Деякі паралельні алгоритми розв’язування задач на власні значення на гібридних комп’ютерах [Електронний ресурс], режим доступу: http//hpc-ua.org/hpc-ua-18/files/proceedings/3.pdf

29. Богурський Д.О. Інтелектуалізація обчислень для задач математичного моделювання складних процесів і об’єктів / В.А. Сидорук, П.С. Єршов, Д.О. Богурський, О.Р. Марочканич // Журнал «Комп’ютерна математика». – 2019. – № 1. – С.143-150., ISSN:2616-938X

30. Богурський Д.О. гібридний алгоритм методу ітерацій для розвязання задач стійкості конструкцій / Д.О. Богурський // Матеріали Міжнародної наукової інтернет-конференції «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення». – м.Тернопіль, 11 червня 2019 р. – С.8-10.

31. Богурський О.Р. Інтелектуалізація обчислень для задач розрахунку стійкості конструкцій / Д.О. Богурський, О.М. Хіміч // Матеріали ІІІ всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 20-22 листопада 2019 р. – С.. – С.75-80.

ДОДАТОК А
ГРАФІЧНИЙ МАТЕРІАЛ

ПЛАКАТ 1 РОЗРАХУНКОВА СХЕМА ЗАДАЧІ СТІЙКОСТІ КОМПОЗИТНОГО МАТЕРІАЛУ

ПЛАКАТ 2 БЛОЧНО-ЦИКЛІЧНИЙ РОЗПОДІЛ МАТРИЦІ

ПЛАКАТ 3 ПРИСКОРЕННЯ ГІБРИДНОГО АЛГОРИТМУ МЕТОДУ ІТЕРАЦІЙ НА ПІДПРОСТОРІ

ПЛАКАТ 4 ЗАЛЕЖНІСТЬ ПРИСКОРЕННЯ ГІБРИДНОГО АЛГОРИТМУ ВІД НАПІВШИРИНИ СТРІЧКИ МАТРИЦІ ТА КІЛЬКОСТІ ВИКОРИСТАНИХ GPU

ПЛАКАТ 5 ЗАЛЕЖНІСТЬ ПРИСКОРЕННЯ ГІБРИДНОГО АЛГОРИТМУ МЕТОДУ ІТЕРАЦІЙ НА ПІДПРОСТОРІ ВІД ПОРЯДКУ БЛОКІВ

ПЛАКАТ 6 СХЕМА ВИКОНАННЯ ОБЧИСЛЕНЬ В ГІБРИДНІЙ ПРОГРАМІ

ПЛАКАТ 7 ПРЕДСТАВЛЕННЯ МАТРИЦІ В РІЗНИХ МАСКАХ КОЛЬОРІВ

ПЛАКАТ 8 СТРУКТУРА НЕЙРОННОЇ МЕРЕЖІ “SPARSE MATRIX VISION”